

Fatten your Mac

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

# Dr. Dobb's Journal

#99 January 1985

\$2.95 (3.50 Canada)

Put 512K  
in your Mac  
for \$300

Use PCDOS's  
new features

Archive files  
automatically  
under CP/M 80  
and CP/M 86



38351 00001

# the closest thing to perfect is WordPerfect by SSI.

Reference Magazine

When it comes to software, nobody's perfect. But according to many of the experts, one word processing program is as close as you can get. No wonder it's called WordPerfect.

What are all the critics raving about?

Simplicity. Most WordPerfect functions require only one keystroke, a simple press of a finger. So you can concentrate on writing, not programming.

Speed. Because it is document-oriented instead of page-oriented, WordPerfect won't make you

wait between pages. No matter how fast you type, WordPerfect won't slow you down.

Features. From writers to doctors, accountants to lawyers, WordPerfect has built-in special functions to meet a wide variety of specific needs. And at SSI, every day is spent upgrading and improving WordPerfect — reaching for perfection.

Get your hands on the critics' choice, WordPerfect word processing from SSI.

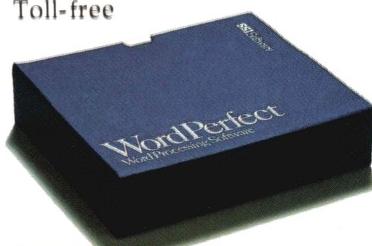
*WordPerfect is my favorite because it is easy, simple and powerful. The people*  
*\* WordPerfect are*

*List Magazine*

It's the closest thing to perfection.

For more information, see your dealer.

Or call or write:  
SSI Software  
288 West Center Street  
Orem, Utah 84057  
Information: (801) 224-4000  
Order Desk: 1-800-321-4566,  
Toll-free



**SSI Software**  
Reaching for perfection.

Digital Review

*WordPerfect isn't flawless word processing software, but it comes very close.*



## Make Whitesmiths, Ltd. Part of Your 1985 Software Strategy.

For six years, software developers and systems integrators have looked to Whitesmiths, Ltd. for technologically superior compilers and multi-tasking operating systems.

Before you make your next move, contact Whitesmiths, Ltd.



Whitesmiths, Ltd.  
97 Lowell Road  
Concord, MA 01742  
TLX 951708 SOFTWARE CNCM.  
(617)369-8499

# Whitesmiths, Ltd.

**DISTRIBUTORS:** Australia, Fawnray Pty. Ltd., Hurstville, (612) 570-6100; Japan Advanced Data Controls Corp., Chiyoda-ku, Tokyo (03) 263-0383; United Kingdom, Real Time Systems, Douglas, Isle of Man 0624-26021; Sweden, Unisoft A.B., Goteborg, 31-125810.

Circle no. 114 on reader service card.

# NO GUTS, NO GLORY.

Parallel port included. Standard interface for popular printers.

Five full-length expansion slots for IBM PC/XT compatible option cards. Hard disk machine with 640K real time clock, three I/O ports and display has two slots free.

Small overall dimensions, sturdy metal case, designed inside and out with horizontal or vertical operation in mind.

**ROM BIOS AND MONITOR** by ITT: Interrupt compatible with IBM PC/XT and more. Ever-ready menu-driven, diagnostic monitor debugger. Test hardware device by device and channel by channel. Examine registers, alter or dump memory, read or write tracks and sectors. Even set drive timing.

256K byte, parity-checked RAM on planar board. Combo board with one, two and three banks of 128K bytes each. Real time clock and parallel port also available to give 640K RAM total with only one slot used.

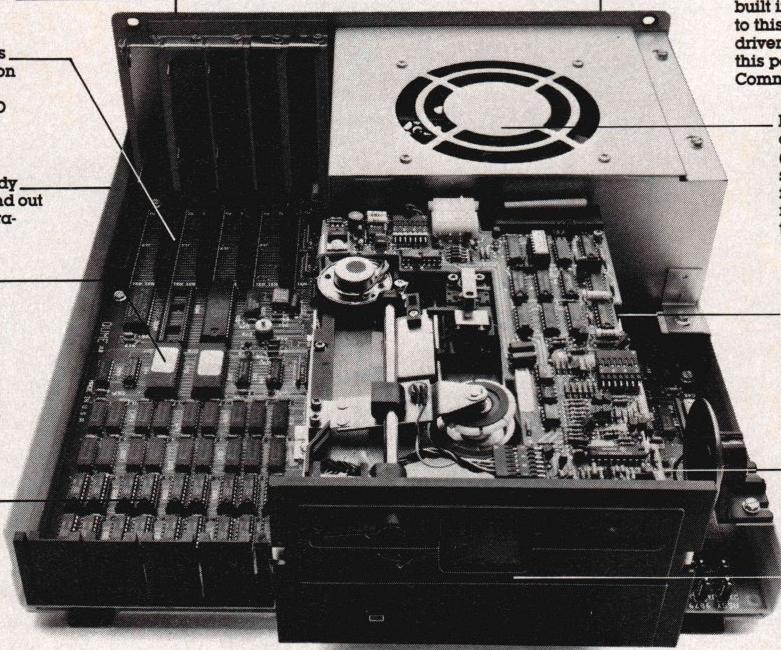
Serial (RS-232) communications port built in. Console I/O may be redirected to this port by switch setting. Menu-driven DOS utility (set up) to configure this port or redirect printer I/O to it. Communications program included.

ITT's own heavy duty, high efficiency, switching power supply (115 watts continuous service) saves weight without compromising support for fully equipped machines. 95 to 132 volts or 180 to 240 volts.

Switch controls allow console I/O redirection to serial port - use whatever ASCII terminal you want. Enable/disable power-on memory test for faster starts on large machines. Enable/disable screen-saving blanking when left idle.

Floppy disk controller on planar. Saves a slot.

Space-saving half-height disk drives.



Monitor conveniently tilts and swivels. Green, amber or color. Three-position keyboard. Our very own, fully-supported ITT Mouse.

Pop the rugged metal case on the new ITT XTRA™ Personal Computer and you'll find what thousands of our personal computer users across the country have already discovered.

The beauty of the ITT XTRA Personal Computer is definitely more than skin deep.

In fact, it runs all the way down to the very last power supply winding.

This is one of the most cleanly designed, most extraordinarily simple machines to come along in quite awhile.

We modestly consider it the finest personal computer compatible on the market.

An example. Where IBM has BIOS support routines for its BASIC, we have a ROM monitor and diagnostics utility that's always available, provides direct disk access and isolates problems right down to the chip level.

It's a finely crafted, expertly engineered machine that's predictable, reliable and a pleasure to use.

We've taken great pains to provide all the right guts.

We leave all the glory to you.

IBM PC and XT are registered trademarks of International Business Machines.

**ITT**  
**PERSONAL COMPUTERS**

## HELPING AMERICA WORK SMART.

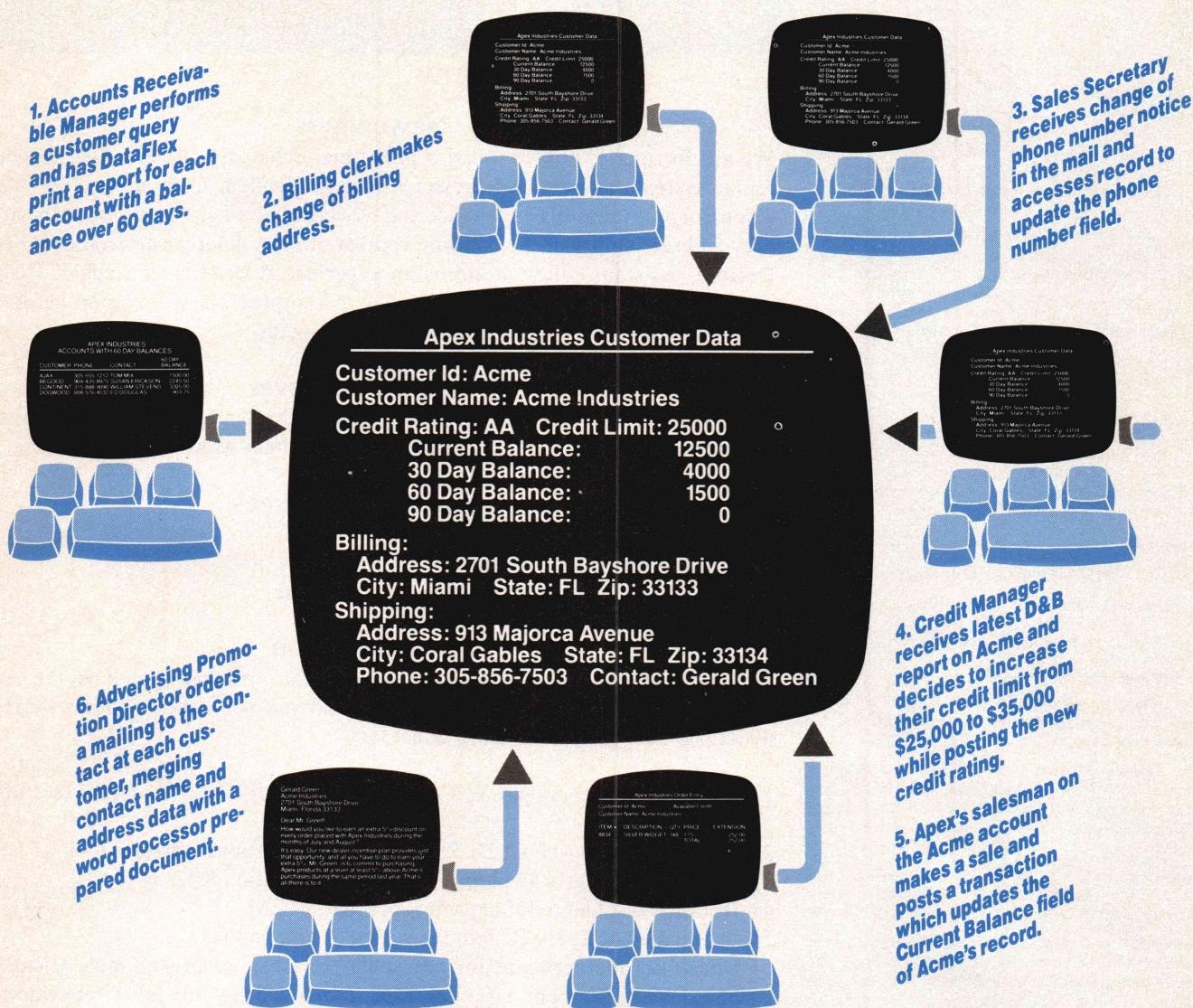
For more information, or the location of your nearest ITT authorized dealer, call 1-800-321-9872.

© 1984, ITT Information Systems

Circle no. 38 on reader service card.

# ALL AT ONCE!

**AND NEVER A "LOCKED OUT" USER!**



DataFlex is the only application development database which **automatically** gives you true multi-user capabilities. Other systems can lock you out of records or entire files for the full time they are being used by someone else. DataFlex, however, locks only the data being changed, and **only** during the micro-seconds it

takes to actually write it to the file! The updated record is then immediately available. The number of users who can access, and change, records at the same time is limited only by the number of terminals on your system or network. Call or write today for all the details on DataFlex...the true multi-user database.

# DATAFLEX™

DATA ACCESS CORPORATION

8525 SW 129 Terrace, Miami, FL 33156 (305) 238-0012  
Telex 469021 DATA ACCESS CI

See us at Comdex Booth 3349

Compatible with CP/M-80, MSDOS networks, MP/M-86, Novell Sharenet, PC-Net, DMS Hi-net, TurboDOS multi-user, Molecular N-Star, Televideo MmmOST, Action DPC/OS, IBM PC w/Corvus, OMNINET, 3Com EtherSeries and Micromation M/NET.

MSDOS is a trademark of Microsoft. CP/M and MP/M are trademarks of Digital Research.

Circle no. 29 on reader service card.

# Dr. Dobb's Journal

## Editorial

Editor-in-Chief *Michael Swaine*  
Managing Editor *Randy Sutherland*  
Assistant Editor *Frank DeRose*  
Contributing Editors  
    *Robert Blum,  
    Dave Cortesi,  
    Ray Duncan,  
    Anthony Skjellum,  
    Michael Wiesenber*  
Copy Editors *Polly Koch; Cindy Martin*  
Typesetter *Jean Aring*  
Editorial Intern *Mark Johnson*

## Production

Design/Production  
    Director *Detta Penna*  
    Art Director *Shelley Rae Doeden*  
Production Assistant *Alida Hinton*  
    Cover *William Cone*

## Advertising

Advertising Director *Stephen Friedman*  
    Advertising Sales *Walter Andrzejewski,  
    Shawn Horst  
    Beth Dudas*  
Advertising Coordinators *Alison Milne,  
    Lisa Boudreau*

## Circulation

Circulation and Promotions Director *Beatrice Blatteis*  
Fulfillment Manager *Stephanie Barber*  
Direct Response Coordinator *Maureen Snee*  
Promotions Coordinator *Jane Sharninghouse*  
Circulation Assistant *Kathleen Boyd*

## M&T Publishing, Inc.

Chairman of the Board *Otmar Weber*  
    Director *C.F. von Quadt*  
    President *Laird Foshay*

Entire contents copyright © 1984 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

**Dr. Dobb's Journal** (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to **Dr. Dobb's Journal**, 2464 Embarcadero Way, Palo Alto, CA 94303. **ISSN 0278-6508**

**Subscription Rates:** \$25 per year within the United States, \$44 for first class to Canada and Mexico, \$62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank.

**Contributing Subscribers:** Christine Bell, W.D. Rausch, DeWitt S. Brown, Burks A. Smith, Robert C. Luckey, Transdata Corp., Mark Ketter, Friden Mailing Equipment, Frank Lawyer, Rodney Black, Kenneth Drexler, Real Paquin, Ed Malin, John Saylor Jr., Ted A. Reuss III, InfoWorld, Stan Veit, Western Material Control, S.P. Kennedy, John Hatch, Richard Jorgenson, John Boak, Bill Spees, R.B. Sutton. **Lifetime Subscribers:** Michael S. Zick, F. Kirk.

**Foreign Distributors:** ASCII Publishing, Inc. (Japan), Computer Services (Australia), Computer Store (New Zealand), Computercollectief (Nederland), Homencomputer Vertriebs GMBH (West Germany), International Presse (West Germany), La Nacelle Bookstore (France), McGill's News Agency PTY LTD (Australia), Progresco (France).

## People's Computer Company

**Dr. Dobb's Journal** is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

**January 1985**  
**Volume 10, Issue 1**

## CONTENTS

## Newcomers

We are fortunate to have found a part time technical editor in Alex Ragen, a senior systems analyst and project leader at Tandem Computers, Inc. Alex's extensive programming experience and writing ability help us to ensure the accuracy and readability of the editorial content of *DDJ*. Another newcomer is Frank DeRose, assistant editor. Frank is an experienced writer and teacher. He was using the Unix system at U.C. Berkeley to compose his dissertation before we lured him away from academia to assist the Doctor.

## This Month's Cover

William Cone, this month's cover artist, used a Macintosh drawing with under-painting in gouache and acrylic to depict this hungry Mac.

## This Month's Referees

Robert Blum, Contributing Editor

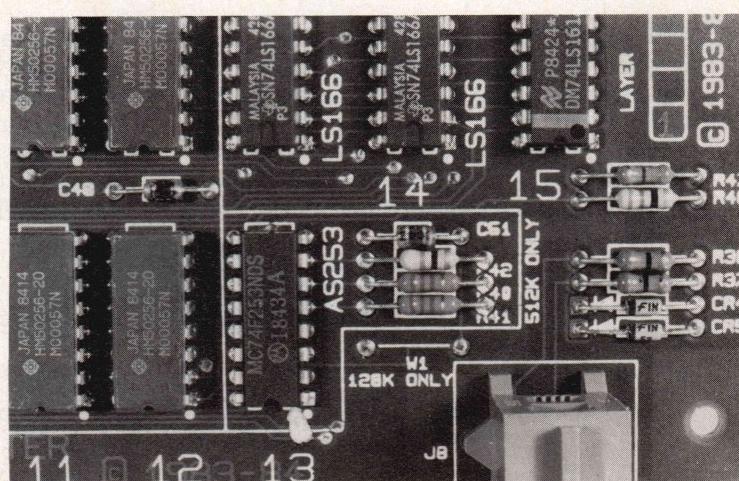
David Cortesi, Resident Intern

Thom Hogan, Editor-in-Chief, Business Software

## Mac Addendum

This arrived too late to make it into the "Fat Mac" article. If you have the newer 128K motherboard, when you get to page 20 you will need to follow these instructions to install the memory select logic.

- 1) Cut and remove jumper W1 (marked 128K only) located at G14 (see photo on this page)
- 2) Install and solder two 2.2K 1/4 w resistors at locations R40 and R41
- 3) Install and solder a 47 ohm 1/4 w resistor at location R42
- 4) Install and solder a .1uf capacitor at location C51
- 5) Install a 74AS253N IC into the socket at location G13
- 6) Check the motherboard for any solder splashes, or broken etch. Clean the board with alcohol or a TG degreaser to remove any flux left after soldering



# Dr. Dobb's Journal

## ARTICLES

- Fatten Your Mac** 18 Step by step instructions to increase the RAM in a Macintosh to 512K; diagrams, photographs, and warnings are included (**Reader Ballot No. 191**)  
*by Thomas Lafleur and Susan Raab*
- QuickDraw Meets ImageWriter** 26 How QuickDraw stores graphics and how ImageWriter prints graphics (**Reader Ballot No. 192**)  
*by Thom Mayer*
- Archiving Files with CP/M 80 and CP/M 86** 36 A utility that determines which files on a disk have been changed and automatically backs them up (**Reader Ballot No. 193**)  
*by Ian E. Ashdown*
- MBOOT and MODEM7 for the C-64's CP/M** 62 All you need to download CP/M software (**Reader Ballot No. 194**)  
*by Walt Piotrowski*
- Unstructured Forth Programming: An Introduction** 86 A modest proposal (**Reader Ballot No. 195**)  
*by Richard Wilton*

## COLUMNS

- Dr. Dobb's Clinic** 12 Review of *A C Reference Manual*, structured search, ragged patches, Z80 assembly riddle (**Reader Ballot No. 190**)  
*by D.E. Cortesi*
- 16-Bit Software Toolbox** 108 PCDOS Version 3.0, graphics routines for IBM PC (**Reader Ballot No. 196**)  
*by Ray Duncan*
- The Software Designer** 124 Interview with Tom Evslin of Solutions Inc.; designing for the Mac: some experiences (**Reader Ballot No. 197**)  
*by Michael Swaine*

## DEPARTMENTS

- |                         |            |   |
|-------------------------|------------|---|
| <b>Editorial</b>        | <b>6</b>   |   |
| <b>Letters</b>          | <b>8</b>   |   |
| <b>Software Reviews</b> | <b>88</b>  | STSC's APL Interpreter, VSI, Fancy Font |
| <b>Book Reviews</b>     | <b>106</b> |   |
| <b>Of Interest</b>      | <b>126</b> | ( <b>Reader Ballot No. 198</b> )        |

# EDITORIAL



I'd been looking forward to this conference for months and wasn't about to let the power failure darken or the rain dampen my spirits. From the cheerful arguments going on around me as we stood dripping in line outside the dining hall, I gathered that others shared my enthusiasm. The organizers of the Hackers' Conference had brought a hundred-odd insanely great programmers and a few writers to this isolated stretch of Marin headlands to discuss the future of the hacker ethic, and discuss the future of the hacker ethic we would; the weather was irrelevant.

The first session dispelled any illusion that we would find consensus on the meaning of the hacker ethic, outside of its having something to do with programming for the joy of it, and nothing to do with breaking into systems. There was a wide span of opinion about the commercial aspect of software development, from the belief that all software should be free to a what-the-traffic-will-bear attitude. Bill Atkinson, admitting that he wanted his QuickDraw routines for the Mac to remain proprietary for a while, looked uneasy when Lee Felsenstein encouraged the attendees to develop, in anarchic concert, a reverse-engineered, non-proprietary Hackers' Mac. Bob Wallace, whose shareware pay-if-you-like-it plan for PC Write was keeping food on his table, seemed amused by the debate. But there was consensus on several significant points: that hacking was worthy in and of itself, that the attendees ought to stay in communication with one another and that it was important to pass the torch to the next generation of hackers. That those present shared something worthy of passing on to others. At the end of the conference I walked out to the ocean and stood on the rocks above the surf, savoring the newfound sense of rational community, as though I had been through a Woodstock for grownups.

Three days later, in Las Vegas, cesspool of the American spirit, I stop midaisle to consult my Comdex program, but a heavy diet of suites has thickened my discriminative faculties until all the booths look alike, nor does the program identify the players, 80 thousand jaded innocents flown in from the undifferentiated heartland to dance the shuck and jive in tight suits, unsensible shoes, flat borrowed jargon. There are, an anonymous passerby pontificates as I pore through the program book, some Interesting Products here this year, though Nothing Revolutionary; this passing summary strikes me as the true smug theme of this year's Comdex. Some fashion war has been won, it seems: the pros, the working press, the veterans, having traded in last year's cynicism for smartmoney conservatism, even the quest for innovation now out of style, pose midaisle, sniffing out the Interesting Products.

One area still open to innovation, I suggest hopefully that evening in some smoky Comdex suite, is software marketing; I'm thinking of shareware, and Bob Wallace, who seems to be everywhere, smiles. The next day I meet Ramon Zamora midaisle in the MGM Grand and he explains that his new company is soliciting grant money to develop shareware, adding that "we call it 'public sector' software."

In another suite Lee Felsenstein elaborates on the philosophy behind the Hackers' Mac. It is a mistake to think of it as a product, he explains; it's a tendency; an appeal to the hobbyist willingness to try things out for no reward but the chance to learn. I promise to mention the idea in Dr. Dobb's.

Back in the office, I check with Randy on the update to the fatten-your-Mac article and the status of our in-house public-domain software bulletin board. Return a call from Bob Albrecht, who's excited about some plan to lure kids into programming in C. Read in Business Week that Steve Jobs thinks "[i]f Apple falters, innovation will cease." And smile.

Michael Swaine

*Imagine  
dBASE III™  
running up  
to 20 times  
faster.*

*The time  
for Clipper  
has arrived.*



## *Clipper introduces you to the time of your life.*

Time is your most valuable commodity. Because how you spend your time, is how you live your life.

At Nantucket, we believe you should live life to the fullest.

Clipper, the first true compiler for dBASE III™ is a timely example. Now, dBASE compiled by Clipper runs 2 to 20 times faster than dBASE with its standard interpreter. A dBASE interpreter painstakingly checks and executes your source code one line at

a time, every time you run a program. With Clipper, once you've debugged your source code, it's compiled into more efficient machine code. Your program runs without the time-consuming overhead of redundant translation. Clipper compiles all your existing and future dBASE III programs.

Developing a compiler for dBASE III was just a matter of time. Call your dealer or our toll free 800 number and ask for Clipper.

Then go make the most of your life time.

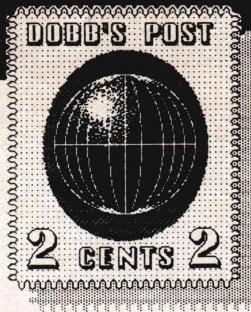


**Nantucket**

20456 Pacific Coast Hwy., Malibu, Ca. 90265 - (800) 556-1234 ext. 225. In California (800) 441-2345 ext. 225

Circle no. 56 on reader service card.

dBASE III is a registered trademark of Ashton Tate



## Spleen Ventilations

Dear Dr. Dobb's,

One very annoying thing that has come up recently is the holy war involving computer languages. It was never more apparent than in your October 1984 issue in the column "The Software Designer." Typical were the statements of Philippe Kahn, who was quoted as saying "C is a disease," and "When I see people writing spreadsheets in C, I think 'They're out of their minds.' It was designed to write operating systems," and "In Europe C is considered an American disease."

Really? C is a disease? How? Why does the fact that C's first job was operating systems disqualify it from spreadsheets? Forth was originally written to manipulate radio receivers. Does this mean that Dr. Dobb's should have rejected the article that demonstrated using Forth for the Fast Fourier Transform? All of Europe hates C? There hasn't been that much unanimity since WWII.

Now it may be true that C has some fatal flaw that will doom it to die, dinosaur-like, in some tar pit of abandoned operating systems. But Kahn certainly hasn't demonstrated it, and he has merely made himself look foolish by making statements that have no correlation with his thesis. And the sad thing is that he is not the only person doing this.

One of the most irresponsible statements I have ever read was that of Edsger W. Dijkstra, in his "How Do We Tell Truths That Might Hurt?" It read: "It is practically impossible to teach good programming to students who have had prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration."

Now, this is patently untrue—some of the best programmers I know start-

ed out self-taught in BASIC and write very clear, structured code. What I assume he meant, in all that rhetoric, was "BASIC is not a good programming language"—which is true in my opinion. But look at that quote again. I'll wager that there were hundreds of mediocre computer science teachers who leapt for joy when they read that. BASIC is implemented on almost every system, which means that it is hard to avoid coming across it and learning it. Having a hard time teaching that student certain programming practices? Well, he must have learned some BASIC—must be he's mentally mutilated. Might as well give up on him; he'll only flunk anyway.

If all we get are diatribes and spleen ventilations, what chance is there for the future of program language design? When a computer scientist merely rants instead of reasons, how seriously can one take claims of the designers of a new language?

It is incredibly annoying reading such nonsense because there are valid and rational reasons for not using a programming language. I personally will not use Forth, because I do not like reverse Polish notation, and because it is too close to assembly language for me to be comfortable with it. But I certainly will not condemn those who use it, and if they write good, efficient programs with it, more power to them.

Perhaps DDJ could run a series of interviews with software designers, discussing what language they use and emphasizing the positive. It might let people see more clearly just what they might be missing.

Thank you for a very interesting magazine.

John M. Gamble  
4471 Eastwood Dr.  
#18103  
Batavia, OH 45103

## grep.c

Dear DDJ,

Thanks for your wonderful public domain contribution of grep.c in the October issue. It has become a well-worn tool in my MSDOS 2.0 programming toolbox. There was an error in the program listing that you might warn your readers about to avoid many hours of "bug" hunting. At the middle of page 61 there is a call to *omatch* contained in a *while* statement passing only two parameters when three are required. The line should be:

```
while (*lin && omatch(&lin,
    pat, boln))
```

Thanks again for the grep.c utility.

Sincerely,  
Michael H. Cox  
3659 Gas Light Curve #5  
Montgomery, AL 36116

Dear DDJ:

It was with interest in learning more about structures and pointers in C that I studied Allen Holub's article on grep.c in the October 1984 issue. Perhaps you would be interested in knowing about an error in Listing Two, page 64. In the *while* loop of the *dodash* module there is a Boolean test on whether *dstart - dest < maxccl*. The last statement of the module is *return (dest-dstart);* Can you tell that one of these is out of order? It turns out that if the former expression is adjusted to read *dest - dstart < maxccl*, then *dodash* will not try to expand a character class to more than *maxccl* characters.

You might think this bug is innocuous, but with my particular setup, this turns out to be fatal to the expansion of sets, like [a - e], to character classes, like [abcde]. I think what's happening is that my compiler sees the pointers *dstart* and *dest* as unsigned, so that

when they are out of order the difference becomes a large unsigned value instead of a small signed value. This makes the *while* stop after the first pass! For the record, I use CI C86 on an IBM PC.

Signed,  
Scott D. Thomas  
354 Colorado Avenue  
Palo Alto, CA 94306

## Complex Numbers

Dear *DDJ*:

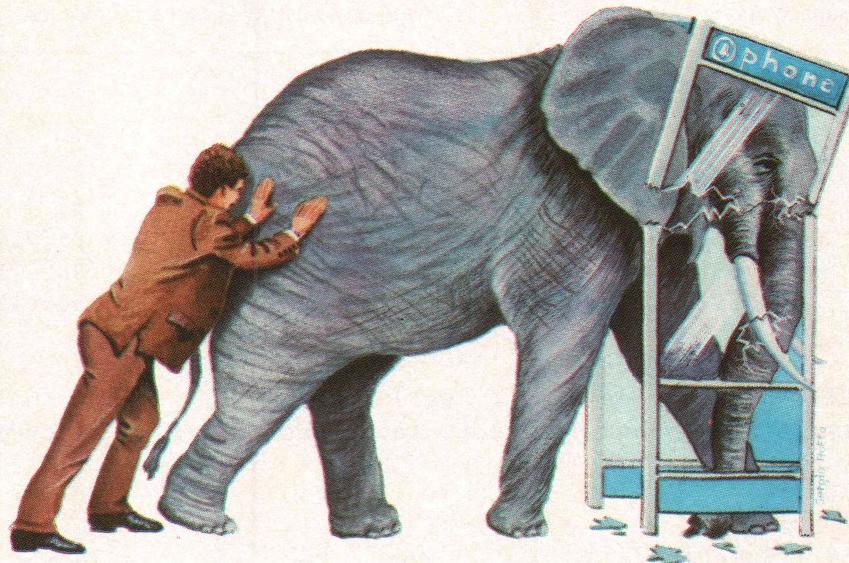
After publication of my article "Simple Calculations with Complex Numbers" in the October 1984 issue of *DDJ*, Mr. Alan Tracht of Cleveland Heights, OH, noticed problems with the routine Polar. For certain arguments, it would return incorrect results. He was kind enough to call me with suggestions for correcting the errors and also told me about a less cumbersome method for preventing floating-point underflow and overflow. A section of the revised program is presented in the Listing (page 10).

The CONSTANT section in the listing is a slightly revised version of the constants from the implementation section of Listing One in the original article. The new version of Polar is much simpler since it passes on most of the hard work to a new FUNCTION called ATan2. This function is a Pascal counterpart to a function found in most Fortran libraries. It takes two arguments, the first a dividend and the second a divisor, and returns the arctangent of the quotient. Because of the extra information inherent in the two arguments, the ATan2 function can correctly handle right angles and determine the correct quadrant of the result. It is tailor made for conversions to polar coordinates.

During testing of the new function, I discovered that the UCSD version of ATan is not quite as "bullet-proof" as I had thought. It was necessary to add some protection for that function as noted in the listing. Such extra protection was not needed by Turbo Pascal or any of the C or Fortran compilers I used to check the new ATan2 algorithm.

The ATan2 function calls another new function, SafeDivide, which attempts to divide its first argument by

# Squeezing A Large Program Into A Small Memory Space?



It's time you got Plink86™ the overlay linkage editor that's bringing modular programming to Intel 8086-based micros.

With Plink86\*, you can write a program as large and complex as you want and not worry about whether it will fit within available memory constraints. You can divide your program into any number of tree-structured overlay areas. 4095 by 32 deep. Work on modules individually. Then link them into executable files. All without making changes to your source program modules.

Use the same module in different programs. Experiment with changes to the overlay structure of an existing program. Use one overlay to access code

and data in other overlays.

Plink86 is a two-pass linkage editor that links modules created by the Microsoft assembler or any of Microsoft's 8086 compilers. Plink86 also works with other popular languages, like Lattice C, C86, or mbp/COBOL. And supports symbolic debugging with Phoenix' Pfix86 Plus™.

Plink86 includes its own object module library manager - Plib86™ - that lets you build libraries from scratch. Add or delete modules from existing libraries...Merge libraries... Or produce cross-reference listings.

Why squeeze any more than you have to? Plink86 by Phoenix. \$395. Call (800) 344-7200, or write.

# Phoenix

**Phoenix Computer Products Corporation**

1416 Providence Highway, Suite 220

Norwood, MA 02062

In Massachusetts (617) 762-5030

\*Plink86 will run under PC DOS, MS-DOS™ or CP/M™-86.

Plink86, Pfix86 Plus and Plib86 are trademarks of Phoenix Software Associates Ltd.

MS-DOS is a trademark of Microsoft Corporation. CP/M is a trademark of Digital Research, Inc.

the second. This function employs the overflow and underflow checking method suggested by Mr. Tracht. It is much less expensive in terms of execution time than the original, which made calls to the natural logarithm function. If anyone has an allergic reaction to the GOTOs, they can be replaced by a copy of the division statement that they jump to. It just takes more space and is redundant.

Finally, I noticed a typographical error I made in the text of the article. In equation 6 on page 31, the calculation of the amplitude is given as taking the arctangent of the real coefficient divided by the imaginary coefficient. In fact, one must take the arctangent of the imaginary coefficient divided by the real coefficient. The order of the dividend and divisor in the original (and new) listing is correct, however.

I hope the erroneous Polar routine did not greatly inconvenience anyone. I would also like to thank Mr. Tracht for his helpful suggestions.

Sincerely,  
David D. Clark  
246 S. Fraser St. #2  
State College, PA 16801

DDJ

## Letters Listing (Text begins on page 8)

```

CONST MAX_REAL = 9.9999999999999999999E+37;
PI = 3.14159265358979323846; { pi }
PI_OVER_2 = 1.57079632679489661923; { pi/2.0 }
BIG_SQRT = 1.0E+19; { Sqrt(MAX_REAL) }
CLOSEST = 1.0E-19; { Sqrt(MIN_REAL) }

FUNCTION SafeDivide(x, y : Real) : Real;
{ divide x by y and intercept incipient underflow or overflow }

LABEL 1;

BEGIN { SafeDivide }
  IF Abs(y) < 1.0 THEN { overflow possible }
    IF Abs(x) > Abs(y)*MAX_REAL THEN
      SafeDivide := MAX_REAL { catch overflow }
    ELSE
      GOTO 1
    ELSE IF Abs(x) < 1.0 THEN { underflow possible }
      IF Abs(y) > Abs(x)*MAX_REAL THEN
        SafeDivide := 0.0 { catch underflow }
      ELSE
        GOTO 1
    ELSE
      1: SafeDivide := x/y { a normal calculation }
END { of SafeDivide };

FUNCTION ATan2(x, y : Real) : Real;
{ take arctangent of x/y and account for right angles and proper quadrant }

VAR a : Real;

BEGIN { ATan2 }
  IF y = 0.0 THEN { check for exceptional cases with y = 0 }
    IF x = 0.0 THEN { if both arguments are zero, return zero }
      a := 0.0
    ELSE
      a := PI_OVER_2 { otherwise it's a right angle }
    ELSE BEGIN
      a := SafeDivide(x, y); { try the divide, preventing unseemly errors }
      IF a >= MAX_REAL THEN { probably tried to overflow }
        a := PI_OVER_2
      ELSE BEGIN
        IF a <> 0.0 THEN BEGIN

```

```

{ kludge to protect UCSD ATan }
a := Abs(a);
IF a > BIG_SQRT THEN
  a := PI_OVER_2
ELSE IF a < CLOSEST THEN
  a := 0.0
ELSE
  a := ATan(a);
END;
IF y < 0.0 THEN      { choose upper or lower half of plane }
  a := PI - a
END
END;
IF x < 0.0 THEN      { choose left or right half of plane }
  a := -a;
ATan2 := a
END { of ATan2 };

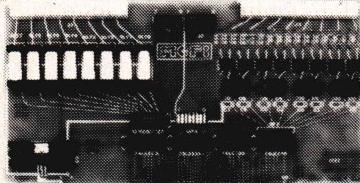
PROCEDURE Polar{Arg: Complex; VAR Modulus, Amplitude: Real};

BEGIN { Polar }
  WITH Arg DO BEGIN
    IF Abs(Re) < CLOSEST THEN
      Re := 0.0;
    IF Abs(Im) < CLOSEST THEN
      Im := 0.0;
    Modulus := Sqr(Sqr(Re) + Sqr(Im));
    Amplitude := ATan2(Im, Re)
  END { of WITH Arg }
END { of Polar };

```

End Listing

## MULLEN S-100: Real Time, Real World CONTROLLER

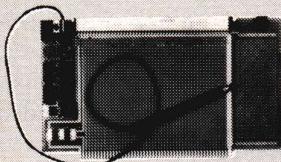


**ICB-10 CONTROLLER BOARD**  
\$219, assembled and tested.

This 8 channel digital I/O controller can monitor inputs and control outputs. It features an easy to read manual that has schematics, component list, and programming examples as well as provocative insights on potential applications.

Examples of applications are included in a reprinted article that demonstrates two MULLEN CONTROLLER BOARDS in an interactive system that: feeds a cat; irrigates a garden dependent on soil moisture; closes the window when it rains; controls the thermostat for optimum comfort; controls appliances, lights, security system, and weather monitoring station (logging temperature, wind speed and direction, and graphing pollution content of the atmosphere.) Solenoids, microswitches, pneumatic actuators, pH sensors, and other devices are used in this system.

## MULLEN S-100:

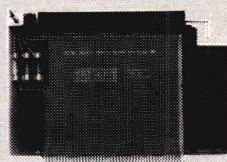


### TB-4a EXTENDER BOARD

The latest in our TB line, the most widely used add-ons in the industry. Features logic probe, formed-lead edge connectors, pulse catcher switch and reset button.

\$89, assembled and tested.

## DEBUGGERS



### ZB-1 ZIF EXTENDER BOARD

This debugger features Zero Insertion Force edge connectors for easy board changes and long life. Expect 2,000 or more insertions rather than the usual 300 to 400 with tension type connectors.

\$159, assembled and tested.



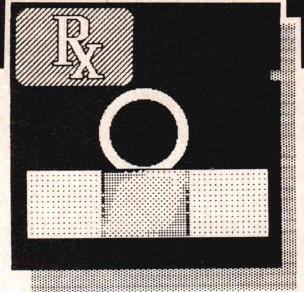
MULLEN COMPUTER PRODUCTS, INC.

### AVAILABLE:

- Priority One Electronics,  
Chatworth, CA • (213) 709-5111
- Jade Computer Products  
Hawthorne, CA • (213) 973-7707
- EI Computer Products  
Hayward, CA • (415) 786-9203

Mullen Computer Products is the industrial distributor for CompuPro's products. For more information, call us at (415) 783-2866 or write MCPI, 2260 American Ave., #1, Hayward, CA 94545. OEM sales available from factory. Prices are subject to change without notice.

Circle no. 52 on reader service card.



by D. E. Cortesi

The compiled and analyzed results of your throughput measurements aren't quite ready for this January column as we promised. Next month.

### We C A Good Book

*A C Reference Manual*, by Samuel P. Harbison and Guy L. Steele, Jr. (Prentice-Hall, 1984; \$19.95) was published just too late to be mentioned in the C Resource List of a few months back. That's a pity because it is the best C resource we've come across yet.

The book is just what its title implies: a complete, authoritative, and (so far as we can tell) accurate reference to the C language. It is beautifully organized, with many small topics grouped logically into chapters. Each topic ends with a list in small type of all the other topics that are related to it, so that no matter where you start you can follow the threads of a concept throughout the book. Differences between C compilers are covered as they occur, matters of coding style and dangerous constructs are discussed, and there are many examples, mostly illuminating.

That it's useful and accurate is a credit to its authors, but that the book is so thorough is the result of its genesis: the authors say it "grew out of our effort to write a family of C compilers." They found that, "In spite of C's popularity . . . there was no description of C precise enough to guide us in designing the new compilers . . . [and none] precise enough for our programmer-customers, who would be using compilers that analyzed C programs more thoroughly than was the custom." So they compiled this one, and a good job they made of it.

### Structured Search

Deep in the code of the DIFF program (presented as a separate article in the

August *DDJ*), we posed a small problem in structured programming. At heart, it was nothing more than the old "loop with two exits," a problem that you're sure to meet whenever you confine yourself to the fundamental control shapes permitted by structured design.

In the context of that program, the problem went like this. A symbol table named ST (an array indexed from 0 to MaxSym-1) is being treated as a hash table to store Lines. Hash (Line) produces the initial probe of the table for any Line. Whenever we probe an entry of the table, we encounter one of three results:

- (1) If ST[S].HashVal is negative, then entry S is free and the present Line may be installed in it.
- (2) If ST[S].HashVal equals Hash(Line) and if ST[S].LineVal equals Line, then this Line has already been entered and S is its index.
- (3) Otherwise, some other Line is hashed to entry S, so we must try the next entry in succession, wrapping around at the end of the table.

The problem fits awkwardly into conventional program structures because the loop must terminate under either case (1) or case (2), but if it terminates under case (1) the new Line must be installed in ST[S]. In the original program (which was hacked together in a hurry), this was all done with Goto statements in an efficient but inelegant way. Several readers responded with rewritten functions, and the variety of their solutions is interesting.

Before looking at them, let's look at a side issue. We stated the equality test in case (2) in two parts for performance reasons. It takes little time to compare two hash signatures (integers). Different Lines, however, will occasionally hash to the same value, in which case (and only then) the slower test of comparing two variable-length strings is applied. In the published pro-

gram, this was handled with a compound IF statement,

```
if (ST[S].HashVal = H)
    and (ST[S].LineVal^ = Line)
    then ...
```

Only Paul Sand of Dover, NH, noticed that such a statement is not good Pascal. "I want to bring to your attention a portability bug," he wrote. "The problem is that standard Pascal does not guarantee short-circuit evaluation of Boolean expressions. For example, in good old Apple Pascal, both tests will *always* be done no matter what the outcome of the first test, slowing things down considerably. Other versions of Pascal might do the tests in reverse order."

Sand is dead right. C promises to short-circuit the second (slower) relation when the first test fails, and Ada offers the CAND (conditional and) operator for explicit control of the sequence of tests. In Pascal, the only way to get the desired effect is to write:

```
if ST[S].HashVal = H then
    if ST[S].LineVal^ = Line then
```

When you do, you end up duplicating code in the ELSE leg of the now doubled IF.

Sand's version of the search code (Listing One, page 17) is a straightforward encoding of the specifications given above. An auxiliary Boolean variable, Done, is introduced to control the loop. Case (1) is handled as soon as it is discovered so that, on termination, cases (1) and (2) have been made identical.

A. Salemma (our approximation from a handwritten signature) of Washington, DC, and Wayne Rivers of Springfield, VA, sent solutions much like Sand's except that they used the more intuitive "Repeat . . . Until

# Hackers, rejoice!

# **Finally, a programmable multi-window editor for MS-DOS.<sup>TM</sup>**

# EMACS

**UniPress EMACS™**

famed Gosling version. Multi-window, text editor with extensibility through the built-in MLISP programming language and macros. Dozens of source code MLISP functions; including C, Pascal and MLISP syntax checking. Run Lattice C or PsMake in the background and Emacs will point to any errors. EMACS now runs on TI-PC, IBM-PC AT, DEC RAINBOW or any MS-DOS machine.

PsMake

**1.5make**  
UNIX style 'make' utility, includes facilities for automatically rebuilding programs based on inter-dependencies of the modules. Includes rule scripts for many popular MS-DOS languages. Also includes many UNIX style tools (ls, cat, touch, etc).

## Lattice C

'the' C compiler for the serious developer. Full C language producing the best code for the 8086 family. All models of 8086 supported. Emacs, PsMake, and Phact all written using Lattice.

Phact Isam

multi-key ISAM for MS-DOS. Uses b+ tree, and supports variable length records. Includes full Lattice linkable library and high-level functions.

**FULL SYSTEM** (*includes Emacs (object), PsMake, Phact, lsam & Lattice C*) — \$1099, with EMACS source, \$1699.

**STANDALONE:** EMACS \$375, with source \$995.

Lattice \$425. PHACT \$250. PSMAKE \$159. One month

EMACS trial \$75.

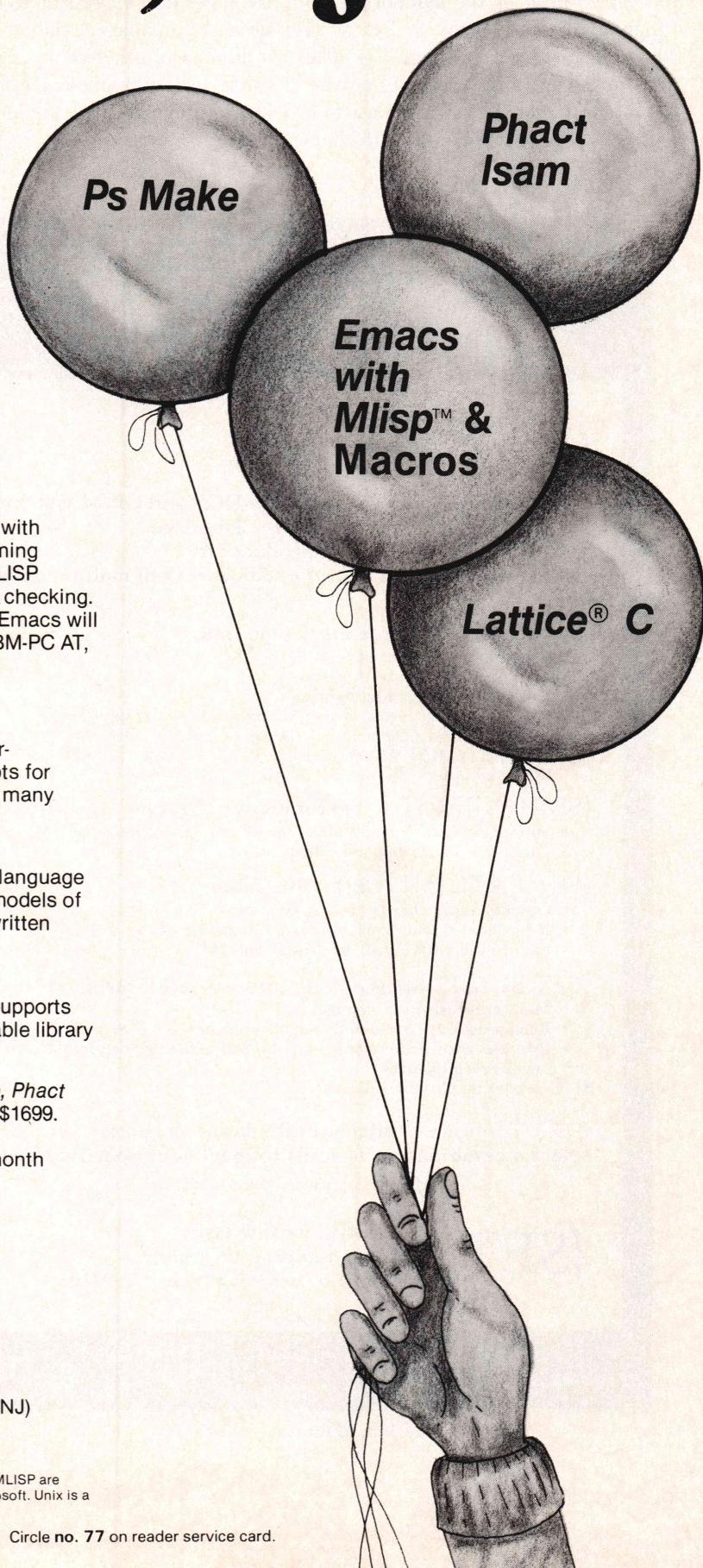
**EMACS PRICE INCREASE FEB. 1  
BE SURE TO PLACE YOUR ORDER**

**BE SURE TO PLACE YOUR ORDER NOW!**

UniPress is a major publisher of U

Call or write for more information.

**UniPress Software, Inc.**  
2025 Lincoln Highway, Edison, NJ 08817  
201-985-8000 • Order Desk: 800-222-0550 (outside NJ)  
Telex: 709418 • Mastercard and Visa  
Japanese Distributor: SofTec Tel: 0480 (85) 6565



Done" structure instead of "While Not Done Do."

It isn't necessary to handle all three conditions inside the loop. Fred Marchand of Bellingham, WA, rewrote the program in C for his own uses, but his implementation of the symbol-search function translates back to Pascal as shown in Listing Two (page 17). The loop ends when either case (1) or case (2) is discovered. The special case (1), entry of a new symbol, is handled out-

side the loop.

Gary Dale of Toronto, Ontario, took an approach which is similar to Marchand's but he made better use of Pascal's treatment of Boolean data (Listing Three, page 17). He was the only one to use a Boolean expression to set the value of the auxiliary variable. Dale made a number of other changes in the program, including implementing his own heap storage so that it could store more lines.

## Ragged Patches

No subject, it seems, inspires more imaginative, not to say panicky, adventures in typography than the program patch presented in a glossy magazine. In the past year, various PC-related magazines have carried articles on patching WordStar for this or that purpose as well as articles on patching other bits of MSDOS or PCDOS; each has used a different way of showing the necessary Debug commands.

It's just nerves, we think. The author knows that making a patch is a risky business; if it isn't done just right, the results will be unpredictable (and, fairly or not, blamed on the author). So the author writes down *exactly* what is to be done, making up typographic conventions for "what the computer will show" and "what the reader should enter" on the fly.

The editors also know that patches are risky things that must be done just right—and that they haven't the expertise to tell which parts of the author's presentation are essential and which are decoration. So they change the author's careful instructions as little as possible; this usually means fitting them to the Procrustean bed of a specific column width, reducing blanks and commas to proportionally spaced insignificance, changing the ASCII apostrophe to an inverted comma, and shrinking fat computer-printed asterisks to eentsy specks above the line.

What's left is a nearly unreadable account of a hypothetical Debug session. Even if it were readable, it wouldn't be relevant. Not everyone uses Debug; other debuggers, and other utilities for modifying disk files, are available. And some patches can be usefully made on the fly by another program. The problem with showing patches in terms of Debug operations is that it confuses the *process* with the desired *results*. All that is relevant about a patch is the address of the target area, what's in that area now, and what to change it to. You can put this data neatly in a three-column table.

The Table on page 17 is an example. It shows a patch for ZDOS version 2.13 (MSDOS 2 for the Zenith 100 series). The patch corrects a tendency of serial output to drop characters. It appeared first in *BUSS, The Independent News-*

### INTRODUCING THE LATEST IN HIGH QUALITY PRODUCTIVITY TOOLS FOR MICROCOMPUTER SOFTWARE DEVELOPERS AND PROGRAMMERS

## {SET} Tools —

- Operate on most popular MS-DOS and CP/M systems.
- Can be used with any source language.
- Improve development productivity.
- Provide assistance for the tedious task of maintenance.

### {SET:DIFS}™ Source File Comparator \$139.00

- Fast, smart and accurate
- Use for regression testing, too
- Difference display highlighting

A/P/L Options available as add-ons to {SET:DIFS} to provide for minimized communications with the ADR or PanValet mainframe librarians or with {SET}'s Batch Line Editor, {SET:LIKE}. \$20.00 per option  
{SET:LIKE} add-on to {SET:DIFS}. \$40.00

### {SET:GXREF}™ Cross Reference Utility \$79.00

- Supplied parameter files allow use of any source language
- Cross references multiple files at once

### {SET:PATCH}™ Object File Editor \$79.00

- Quickly apply changes to any file type
- Hexadecimal and ASCII display and change entry
- Easy to use with cursor and function keys

### {SET:SCIL}™ Source Code Interactive Librarian \$349.00

- Maintains history of changes
- Reduces storage by identifying differences from level to level
- Provides control over concurrent development efforts by detecting overlapping changes

PC-Demo Disk and Manual available for \$35.00

{SET} Tools are available individually or, better yet, select a combination of tools to meet your specific needs.

Multiple copy discount available.

**{SET}** Get {SET} for Success  
System Engineering Tools, Inc.  
645 Arroyo Drive • San Diego, CA 92103

COD, Check with order, Master Card or VISA accepted.

To order {SET} tools or for more information, call  
System Engineering Tools, Inc. (619) 692-9464.

Circle no. 69 on reader service card.

# How to earn an MBA in one day.

LIMITED EXHIBIT  
SPACE AVAILABLE  
CALL 800-OAC-1985

## MASTER OF BUSINESS AUTOMATION.

This certifies that

*John Doe*

has successfully completed the requirements for  
MASTER OF BUSINESS AUTOMATION  
by attending OAC '85 at the Georgia World Congress Center.  
Atlanta, Georgia.

Having visited the world's largest exhibit and conference  
on Business Automation, the above named is an expert on  
the ways Business Automation can improve productivity,  
efficiency, and profitability.

OAC '85 February 4-6, 1985

*Alfred Thomas*      *John Doe*



## Attend OAC '85, February 4-6. And master the Business Automation revolution.

This is your only chance to see the largest, most extensive business automation exhibit and conference in the world.

February 4-6, Georgia World Congress Center, Atlanta Georgia.

### See 150 major exhibitors including:

|                               |                               |
|-------------------------------|-------------------------------|
| Apple Computer Inc.           | Honeywell Information Systems |
| AT&T Communications           | IBM Corporation               |
| AT&T Information Systems      | Lanier Business Products      |
| Bell South Services           | Minolta Corporation           |
| Bell & Howell                 | McGraw Hill                   |
| Burroughs Corporation         | NCR Corporation               |
| Data General                  | NEC Information Systems, Inc. |
| Dictaphone Corporation        | Radio Shack                   |
| Digital Equipment Corporation | Sperry                        |
| Dupont Company                | Wang Laboratories             |
| Eastman Kodak                 | Xerox Corporation             |
| Hewlett-Packard Co.           |                               |

### It's office automation and much more:

- Complete Conference Program with over 48 sessions in 6 program tracks:
- organizational impact of technology • office workstations
- ergonomics of the workplace • networking applications
- communications technologies and issues
- productivity and requirements evaluation
- Conference Keynote-Howard Anderson, Managing Director of the Yankee Group.
- 12 in-depth Professional Development Seminars
- The latest in Automation, Communications and Integration, including telecommunications and software
- Micro to mainframe connection

Get in the forefront of the  
Business Automation  
revolution. Order your  
registration information now.

## Call 800-OAC-1985

I can't afford to miss OAC '85. Please  
rush me my registration information.

Name: \_\_\_\_\_

Title: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

City: \_\_\_\_\_ State: \_\_\_\_\_ Zip: \_\_\_\_\_

Mail to:  
OAC 85 AFIPS 1899 Preston White Dr. Reston, VA 22091 GI

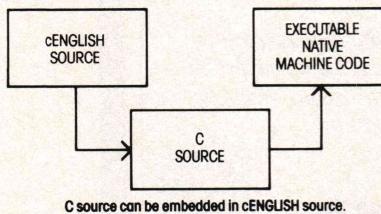
OAC  
'85  
THE WORLD'S  
BEST

# cENGLISH.<sup>TM</sup>

## The C Generation Language.

**What is cENGLISH?** cENGLISH is a comprehensive fourth generation procedural language based on dBASE II<sup>®</sup> syntax. It is portable to a wide range of micros and minis. The language features user-transparent interfaces to a wide range of popular C compilers, operating systems, and data base managers.

**How is portability achieved?** cENGLISH through its compiler interface translates cENGLISH into documented C source and uses a host C compiler to produce native machine code.



Differences in the operating system and data base manager are handled by the runtime libraries.

The result is that cENGLISH source can be compiled without modification on any micro or mini configuration supporting cENGLISH.

**What about performance?** cENGLISH executes FAST, just like any compiled C program.

**How easy is cENGLISH to use?** While cENGLISH is a powerful high level language that can accommodate complex software development, it remains simple and straightforward to use.

**Call or write for availability of cENGLISH for the following configurations—**

Compilers:

Standard O/S compilers: Lattice C™ for MS/DOS™

Operating Systems:

UNIX,<sup>®</sup> UNIX-like, MS/DOS,<sup>™</sup> Coherent,<sup>™</sup> VMS<sup>™</sup>

Data Base Managers:

C-ISAM<sup>™</sup> and INFORMIX,<sup>™</sup> UNIFY,<sup>™</sup> ORACLE,<sup>™</sup> PHACT,<sup>™</sup> Logix<sup>™</sup>

Foreign Language Versions:

German, French, Spanish

**Attention MS/DOS users.** Demo version and special introductory offer available for IBM PC,<sup>™</sup> XT,<sup>™</sup> AT,<sup>™</sup> and other MS/DOS systems.

Requirements: 256K, hard disk or two floppy disk drives, and MS/DOS 2.1 or higher.

**Attention dBASE II and dBASE III users.** dBASE II to cENGLISH Converter now available; dBASE III Converter available later this quarter. Converted code is portable to micros or minis and executes as fast as original cENGLISH source.

dBASE II and dBASE III are trademarks of Ashton-Tate. Lattice is a trademark of Lattice, Inc. UNIX is a trademark of Bell Laboratories. MS/DOS is a trademark of Microsoft, Inc. Coherent is a trademark of Mark Williams Company. VMS is a trademark of Digital Equipment Corporation. C-ISAM and INFORMIX are trademarks of Relational Database Systems, Inc. Oracle is a trademark of Oracle Inc. PHACT is a trademark of Phact Associates. Logix is a trademark of Logical Software, Inc. IBM PC XT and AT are trademarks of International Business Machines Corporation. UNIFY is a trademark of Unity Corp.



### SAMPLE cENGLISH PROGRAM

#### IDENTIFICATIONS

MODULE: Mininame  
AUTHOR: bcs  
DATE: 8/29/84  
REMARKS: Sample cENGLISH program that adds first names to a file  
END IDENTIFICATIONS

#### Globals

FIXED LENGTH 1 ans  
FIXED LENGTH 15 Fname  
END GLOBALS

#### MAIN PROGRAM

BEGIN  
CLEAR SCREEN  
SET ECHO OFF  
USE "NAMES"  
VIEW BY "ID\_FNAME" ASCENDING

AT 23,1 SAY "Add a record? Y or N"  
AT 23,25 ENTER ans USING "

WHILE ans EQ "Y"  
CLEAR GETS  
AT 6,1 SAY "Enter first name"  
AT 6,20 GET Fname  
READ SCREEN

INSERT  
Fname = Fname  
END INSERT

AT 12,10 SAY "Welcome to cENGLISH," & Fname  
WAIT  
AT 14,10 SAY "HIT ANY KEY TO CONTINUE"  
STORE " " TO Fname  
STORE " " TO ans  
AT 23,1 SAY "Add another record? Y or N"  
AT 23,30 ENTER ans USING "  
CLEAR ROW 1 THRU 23

END WHILE

AT 12,10 SAY "That's all for now!"  
UNUSE "NAMES"  
SET ECHO ON

END PROGRAM

I'd like to know more about cENGLISH.  
Please send further information.

Your Name \_\_\_\_\_ Title \_\_\_\_\_

Company \_\_\_\_\_ Telephone \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_  
Check one:  End User  System House  Dealer  Distributor

Send to: CLINE Inc., 20 West Ontario, Chicago, IL 60610-3809

Telex 516315 Phone (312) 944-4510

In Canada: CLINE Canada, Inc. Complexe La Laurentienne,  
425 St. Amable, Suite 165, Quebec, Canada G1R5E4  
Phone (418) 524-4641

FQ185

*letter of Heath Co. Computers* as a column and a half of tortuous instructions. The table is clearer and less likely to lead to error. It informs the reader what is to be changed, but leaves the how of the change unspecified.

(Charles Floto's *BUSS*, incidentally, is a valuable resource for anyone who owns Heath hardware. It costs \$28 for a year of 20 gossip- and bargain-filled issues; call (202) 544-0900 to subscribe.)

## Wat Duzzit Dew?

David S. Tilton sent us the following sequence of Z80 assembly code. It's an absolutely astonishing implementation of... but you can figure it out. Unfortunately, it requires such narrowly defined conditions that it's probably useless, despite being the fastest one of its kind. What does it do, and what are its narrow requirements for utility?

|       |      |
|-------|------|
| Loop: |      |
| cp    | (hl) |
| ret   | z    |
| rl    | L    |
| djnz  | Loop |
| ld    | L,O  |
| ret   |      |

Might there be a way to use this gimmick in another architecture (e.g., 8086, 68000) to get the same result with more flexibility? **DDJ**

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 190.

Load IO.SYS with the debug command: L 1000:0 0 A 20

| At ...    | Find ...   | Replace ... |
|-----------|------------|-------------|
| 1000:1E32 | MOVE AH,AL | MOV BH,AL   |
| 1000:1E47 | POP BX     | MOV AL,BH   |
|           | MOV AL,AH  | POP BX      |
| 1000:1E91 | MOV AH,AL  | MOV BH,AL   |
| 1000:1EA6 | POP BX     | MOV AL,BH   |
|           | MOV AL,AH  | POP BX      |

**Table**

**A patch to the BIOS of ZDOS 2.13 (not PCDOS) to prevent dropping bytes in serial output. Addresses based on loading IO.SYS from disk under Debug; not correct for patching the active system.**

## Dr. Dobb's Clinic

(Text begins on page 12)

### Listing One.

Structured hash-table search by Paul Sand

```

h := hash(Line);
s := h mod MaxSym;
done := FALSE;

while not done do
  if (ST[s].HashVal < 0) then begin
    with ST[s] do begin
      hashval := h;
      new(LineVal);
      LineVal^ := Line
    end;
    done := TRUE
  end
  else
    if ST[s].HashVal = h then
      if ST[s].LineVal^ = Line then
        done := TRUE
      else
        s := (s + 1) mod MaxSym
    else
      s := (s + 1) mod MaxSym;
store := s; { result }

```

**End Listing One**

### Listing Two.

Structured hash-table search by Fred Marchand

```

h := Hash(Line);
s := h mod MaxSym;
found := FALSE;

while (ST[s].HashVal > 0) and (not found) do
  if (ST[s].HashVal = h) and

```

```

(ST[s].LineVal^ = Line) then
  found := TRUE
else
  s := (s+1) mod MaxSym;

```

```

if (ST[s].HashVal < 0) then
  with ST[s] do begin
    HashVal := h;
    new(LineVal);
    LineVal^ := Line
  end;

```

store := s;

**End Listing Two**

### Listing Three.

Structured hash-table search by Gary Dale

```

h := Hash(Line);
s := h mod MaxSym;

repeat
  with ST[s] do
    if (HashVal = h) then
      found := LineVal^ = Line
    else
      found := HashVal < 0;
    if not found then
      s := (s+1) mod MaxSym
  until found;

if (ST[s].HashVal < 0) then
  with ST[s] do begin
    HashVal := h;
    new(LineVal);
    LineVal^ := Line
  end;

```

store := s;

**End Listings**

*The modifications described in the article by Lafleur and Raab are not for amateurs. In fact, we can't with a clear conscience recommend that you void your warranty and risk destroying your motherboard to save a few dollars. Although we've taken steps to convince ourselves of the technical accuracy of the article (and we know that Tom Lafleur has used this procedure on several Macs), we haven't yet had the nerve to fatten a Mac ourselves. Don't take this on as a first hardware project. Don't undertake it if you aren't sure of the risks involved. And don't blame us if anything goes wrong. We are providing this information strictly as a service to those who know how to use it. We take no responsibility for fried Macs. If you have the newer 128 motherboard, read the addendum on page 4.*

**A**re you tired of those pesky disk writes while using MacPaint? Does the performance of even your new hard disk drive leave you longing for the lightning response of a RAM disk? Want to run Lotus 1-2-3 and can't, or are you developing MAC software and just plain running out of room?

For these and many other reasons, Macintosh owners everywhere are rushing to Apple for a FAT MAC upgrade that packs a full 512K of memory into the system. Most will (and should) wait for Apple to add the necessary chips to the Macintosh motherboard. However, if you are one of the adventurous few willing to sacrifice your Apple warranty and wager the life of your Macintosh against your soldering skills, here's how to perform the upgrade yourself for half the cost.

#### **You'll Need... .**

The checklist in Table I (page 21) shows the parts and tools you'll need to complete the job. To minimize your MAC's down time, make sure you have everything on the list before you begin. All of the parts can be found at your local electronics supply house or by looking for ads in your favorite computer magazine.

Because the 256K RAM chips are a little harder to find, I have listed a few sources. We have paid \$15 to \$45 each for the memory chips: they are not cheap! We've tested only NEC memory chips in this conversion, but other equivalent 256K memory chips should work as well. Once you've got the parts and equipment together, you're ready to begin.

#### **The Procedure**

You'll need about four hours of quiet concentration to complete this upgrade yourself. Before you begin, read through the entire procedure. Then disconnect all cables from the Macintosh, including the keyboard, mouse, printer, extra drive, power, and anything else you might have plugged in. (1) **Remove the case screws.** Use the Xcelite XTD-10 Torx screwdriver to remove all five of the case screws (see Photo 1 on page 20). There are two near the bottom, one inside the battery case, and two deep inside the handle.

---

*Tom Lafleur, P.O. Box 490, Del Mar, CA 92014.*

*Susan Raab, Digital Research International, 160 Central Avenue, Pacific Grove, CA 93950.*

**(2) Remove the case.** Turn the Macintosh facedown on a table. Gently press down on the battery compartment, power connector, and I/O connectors while you lift up on the back of the case. The back of the case should come off in your hands, leaving the faceplate in position around the screen and disk drive door. If the case is stubborn, insert a long metal ruler into the seam between the faceplate and the back of the case (see Photo 2, page 20). (*Do not* use a screwdriver here.) Use the ruler to gently pry the seam open as you pull up on the back of the case.

**(3) Remove the motherboard.** Find and disconnect the two cables that attach the display board and disk drive to the motherboard (see Photo 3, page 20). Ease the motherboard from its connections and remove it from the chassis. Check the revision number of the motherboard. We've successfully performed this upgrade on motherboards that have the revision numbers 630-0101 screened on top and 820-0086C etched on the back.

**(4) Locate and remove the memory chips.** You'll find sixteen memory chips with part number MT-4264 at IC locations F5 through F12 and G5 through G12 (see Photo 4, page 20). Remove them from the motherboard as follows:

- Using a small pair of wire cutters, clip all the pins off each chip as close to the chip as you can (see Photo 5, page 20). Then throw away all 128K of these memory chips—they aren't as valuable as the Macintosh motherboard!

- Using a low-temperature soldering iron (700°), remove all the memory chip pins from the motherboard (see Photo 6, page 21). It's handy to use a Weller WTCPN soldering station because it has a magnetic tip that can pull the pin out of the board as soon as the pin is heated, but other soldering irons will work if you use care. This is a repetitive task (256 pins!) and you'll soon develop a rhythm. To keep the rhythm going, skip over pins 8 and 16 from each chip on the first pass. Because they're connected to the inner power and ground layers of the motherboard, they

require more heat and time to remove. If you remove them on a second pass, they'll establish their own rhythm.

- Clean out all the holes with a solder sucker (see Photo 7, page 21). Again skip holes 8 and 16 on your first pass to keep your rhythm going, then clean them out on a second pass.

- Clean the motherboard with a fine brush. Look the board over for any solder splashes or broken etch.

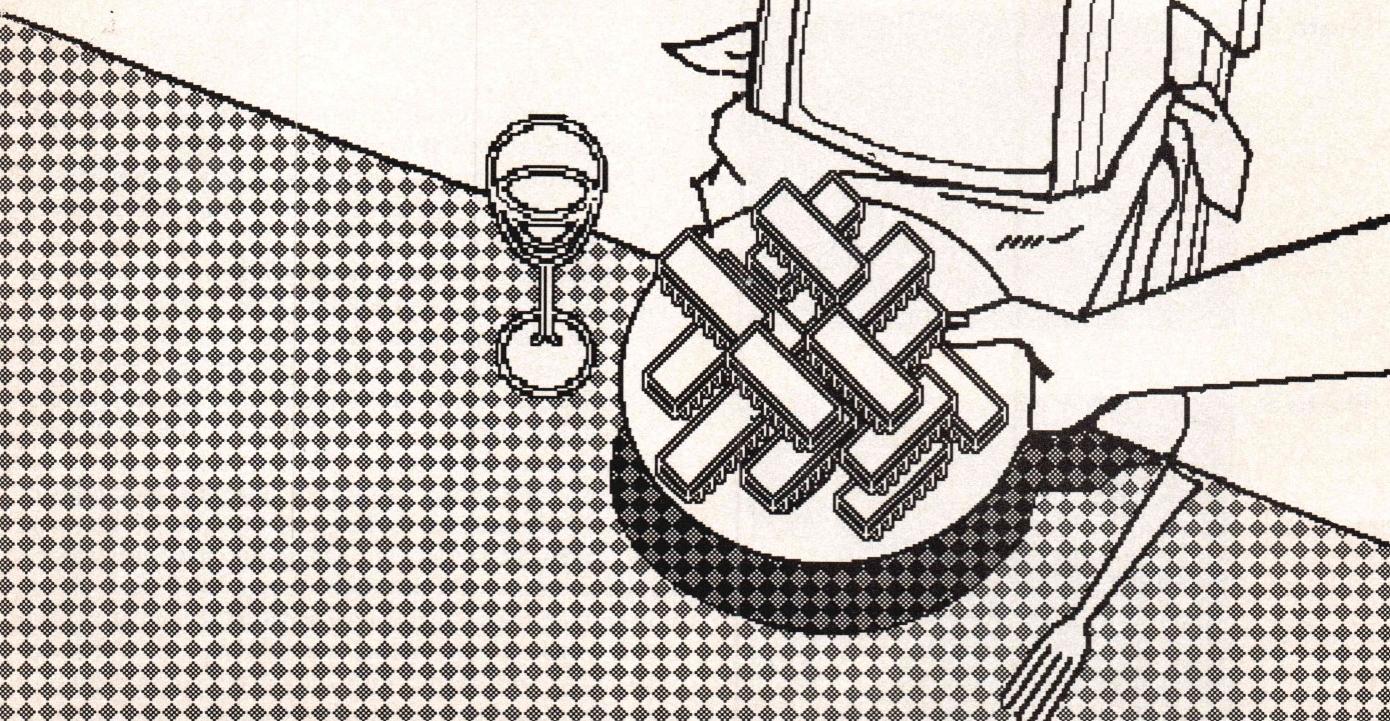
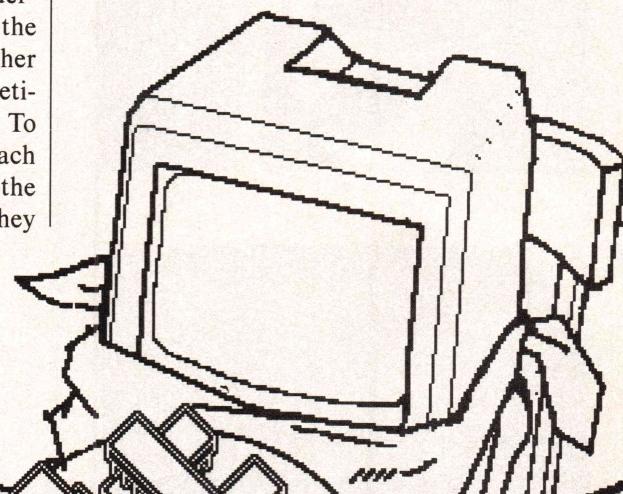
**(5) Insert IC sockets.** Carefully solder a good IC socket in each memory chip location. When you're done, clean the back side of the motherboard with alcohol or a TF degreaser to remove all of the flux left after soldering. Examine the motherboard again for short or broken etch.

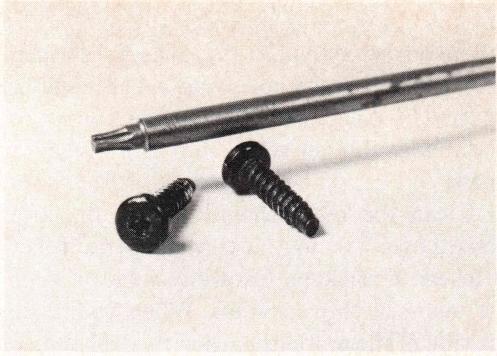
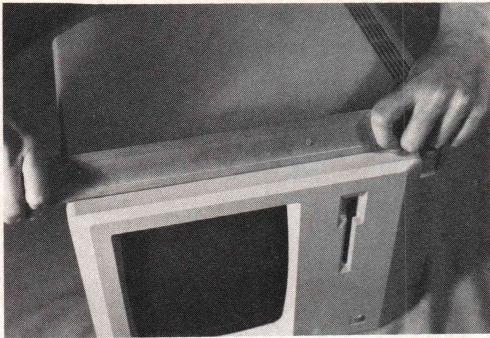
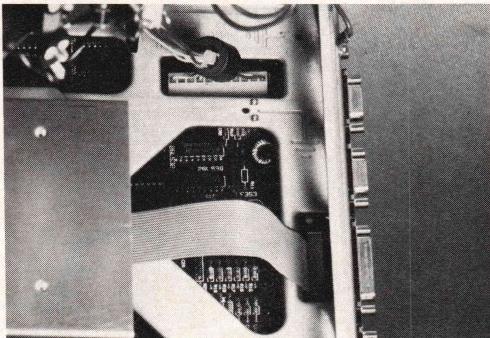
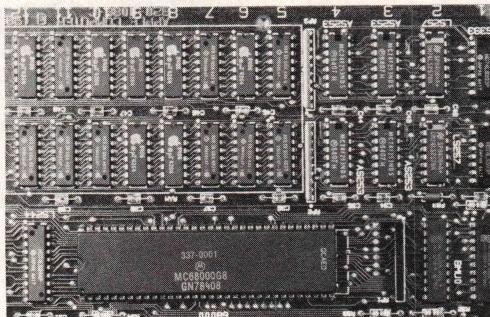
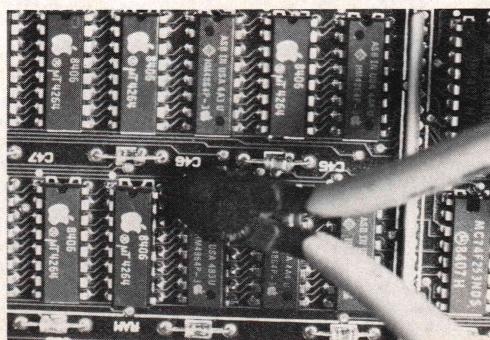
**(6) Install new memory chips.** Carefully insert the sixteen 41256 memory chips into the sockets (see Photo 8, page 21).

**(7) Test motherboard.** Before you go any further, check your work by starting up the system. At this point, it should operate as a normal 128K Macintosh. To complete the test, follow these steps:

- Insert the motherboard back into the Macintosh chassis. Connect the display and drive cables to the motherboard, then connect the power cable.

- Start up the system. If all is well, you'll see the normal question-mark disk icon. If the system's diagnostic software finds a problem, it will display a sad icon (or as much of one as it can!). If the system does not start nor-



**Photo 1****Photo 2****Photo 3****Photo 4****Photo 5**

mally, check the trouble-shooting section at the end of this article. *Do not go on to the next step until the system boots up normally.*

(8) **Remove the motherboard.** Again disconnect the display, disk, and power cables from the motherboard. Ease the motherboard from its connectors and remove it from the chassis.

(9) **Assemble a new memory select IC.** You must build a memory select IC that lets the Macintosh access the extra memory you've just installed. Create the new IC as follows, using Photos 9 and 10 (page 21) and Figures One and Two (page 23) for reference.

- On a good quality 16-pin IC socket, bend out all the pins except 2, 7, 14, and 16.
- Use solder and some small-gauge wire (30 awg) to connect pins 1, 10, 11, 12, and 13 to pin 8.
- Solder a 2.2K 1/4 watt resistor between pins 15 and 16.
- Use solder and some small-gauge wire to connect pins 3 and 4 to pin 15.
- Insert a 74F253 or a SN74AS253 dual 4-to-1 multiplexer into this modified IC socket.

(10) **Install the new memory select IC.** You must connect the new IC assembly to the 68000 address bus, memory select logic, and other lines on the motherboard as follows:

- Mount your new IC assembly on top of the 74F253 (or SN74AS253) located at F3 on the motherboard. Solder pins 2, 7, 14, and 16 of the new IC assembly to the same pins on the 74F253 at F3.
- Locate the seven IC pads at E3, next to pins 32 and 33 of the 68000 microprocessor (see below):

|       |    |                     |
|-------|----|---------------------|
| 68000 | 32 | 33                  |
|       | 1  | o      (+ 5 Volts)  |
|       | 2  | o      (Dram pin 1) |
|       | 3  | o      (A18)        |
|       | 4  | o      (SEL - A)    |
|       | 5  | o      (SEL - B)    |
|       | 6  | o      (A17)        |
|       | 7  | o      (GND)        |

32

- On the back side of the motherboard, cut the etch between pins 1 and 2 at location E3.
- Solder a 47-ohm 1/4 watt resistor between pin 7 of the new IC assembly and pin 2 of the IC pad at location E3. (See Figure One.)
- Using solder and small-gauge wire, connect pin 5 of the new IC assembly to pin 3 of the IC pads at location E3.
- Using solder and small-gauge wire, connect pin 6 of the new IC assembly to pin 6 of the IC pads at location E3.
- Check the motherboard for any solder splashes, missed wiring, or broken etch. Clean the board with alcohol or a TF degreaser to remove any flux left after soldering.

(11) **Test your FAT MAC!** Insert the upgraded motherboard into the Macintosh chassis. Reconnect the disk, display, and power cables. Power up your Macintosh and check for the normal question-mark disk icon. If your MAC does not appear normal, review the trouble-shooting section at the end of this article. If the normal question-mark disk icon appears, insert your system disk and open the disk copy program. The disk copy program should display a message that

says it only works with a standard 128K MAC. To see how much memory you have, start up BASIC (if you have it) and enter the free memory command: PRINT FRE(0). On our system, BASIC reports about 340K of memory. You may also want to run the MAC memory diagnostics as outlined below for a few hours to check for any long-term problems.

### Trouble Shooting

If the system does not start normally, use a multimeter to check that all the connections you've made are connected properly. If all the connections check out, use the system diagnostic program in the Macintosh ROM to determine if

#### Parts:

| Qty. | Part No.                    | Description   | Vendor                    |
|------|-----------------------------|---|---------------------------|
| 16   | 41256-200                   | 256K 200-ns<br>memory chips   | NEC                       |
| 17   |                             | 16-pin IC sockets   |                           |
| 1    | SN74AS253N<br>or<br>74F253N | Dual 4-to-1 multiplexer   | TI, Motorola<br>Fairchild |
| 1    |                             | 2.2K 1/4 watt resistor<br>(any resistor from 1K<br>to 4.7K will work) |                           |
| 1    |                             | 47-ohm 1/4 watt<br>resistor   |                           |

#### Memory chip vendors:

The first five vendors sell their products retail and have a small or no minimum order; the other vendors are industrial suppliers and may have a minimum order.

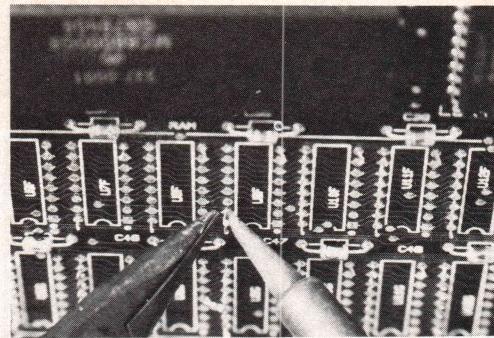
|                           |                |
|---------------------------|----------------|
| Jameco electronics        | (415) 592-8097 |
| JDR microdevices          | (408) 995-5430 |
| Advance computer products | (800) 854-8230 |
|                           | (714) 558-8822 |
| Jade computer products    | (800) 421-5500 |
|                           | (800) 262-1710 |
| DoKey computer products   | (800) 538-8800 |
|                           | (800) 848-8008 |
| NARA                      | (408) 748-9200 |
| TAKA                      | (415) 952-9000 |
| Japan electronics         | (818) 369-1833 |
| D-L-C                     | (213) 938-2677 |

#### Tools:

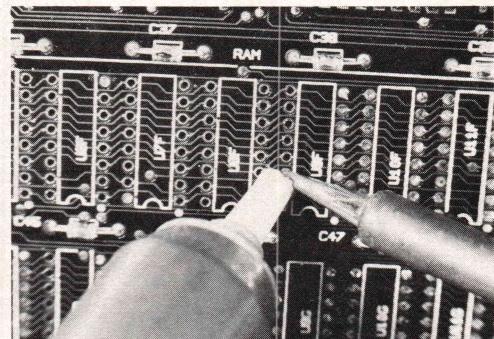
- Xcelite XTD-10 Torx screwdriver, 6-inch shaft
- Long metal ruler
- Low-temperature soldering iron  
(Weller WTCPN recommended)
- Solder sucker
- Small-gauge wire (30 awg)
- Multimeter (for problem solving)

**Table I.**  
**Supplies Checklist**

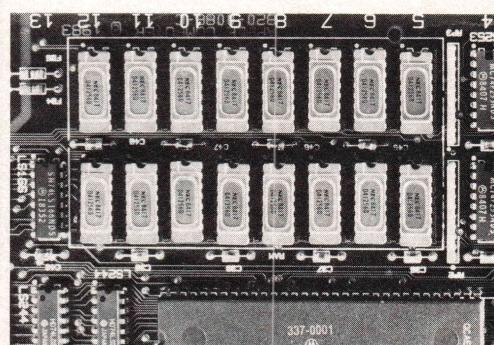
**Photo 6**



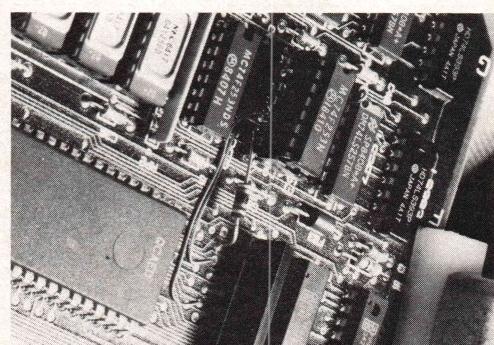
**Photo 7**



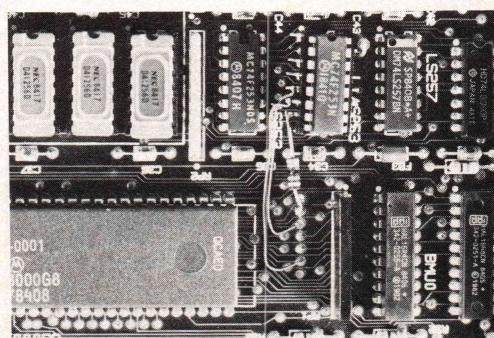
**Photo 8**



**Photo 9**



**Photo 10**



| Chip Location | Data Bit Number | Pin 2 to 68000 Pin | Pin 14 to Memory Buffer Pin |
|---------------|-----------------|--------------------|-----------------------------|
| F5            | D0              | 5                  | E12 - 2                     |
| F6            | D1              | 4                  | 4                           |
| F7            | D2              | 3                  | 6                           |
| F8            | D3              | 2                  | 8                           |
| F9            | D4              | 1                  | 17                          |
| F10           | D5              | 64                 | 15                          |
| F11           | D6              | 63                 | 13                          |
| F12           | D7              | 62                 | 11                          |
| G5            | D8              | 61                 | E13 - 2                     |
| G6            | D9              | 60                 | 4                           |
| G7            | D10             | 59                 | 6                           |
| G8            | D11             | 58                 | 8                           |
| G9            | D12             | 57                 | 17                          |
| G10           | D13             | 56                 | 15                          |
| G11           | D14             | 55                 | 13                          |
| G12           | D15             | 54                 | 11                          |

**Table II.**  
**Connections for Pins 2 and 14**

| Class Code                           | Sub Code             |
|--------------------------------------|----------------------|
| 1 = ROM test failed                  | Meaningless          |
| 2 = Memory test - bus subtest        | Identifies bad chips |
| 3 = Memory test - byte write         | Identifies bad chips |
| 4 = Memory test - Mod3 test          | Identifies bad chips |
| 5 = Memory test - address uniqueness | Identifies bad chips |

**Table III.**  
**Diagnostic Codes**

| Data Bit | Location | Sub Code Bits |
|----------|----------|---------------|
| 0        | F5       | 0001          |
| 1        | F6       | 0002          |
| 2        | F7       | 0004          |
| 3        | F8       | 0008          |
| 4        | F9       | 0010          |
| 5        | F10      | 0020          |
| 6        | F11      | 0040          |
| 7        | F12      | 0080          |
| 8        | G5       | 0100          |
| 9        | G6       | 0200          |
| 10       | G7       | 0400          |
| 11       | G8       | 0800          |
| 12       | G9       | 1000          |
| 13       | G10      | 2000          |
| 14       | G11      | 4000          |
| 15       | G12      | 8000          |

**Table IV.**  
**Chip Identification**

any of the memory chips you've inserted are bad.

Checking the connections is another repetitive task: you must make sure that all of the 256 new connections you've made carry signal to the appropriate destinations. And most of the connections carry signal to more than one place!

For example, a signal on pin 0 on one memory chip should be connected to pin 0 on every other memory chip. A signal on pin 1 should appear on pin 1 of all the other memory chips. Check for this continuity on all pins except 2, 14, and 15.

Pin 15 is common among chips in the same row. For example, pin 15 on a chip in row F should be connected to every other pin 15 in row F but not in row G. Pin 15 on a chip in row G should be connected to every other pin in row G. Pin 2 on each of the sixteen memory chips is directly connected to one of the sixteen data lines of the 68000 microprocessor. Pin 14 connects the memory chips to the memory buffer circuits at locations E12 and E13.

Table II (at left) shows how pins 2 and 14 should be connected. Each row in the table gives information about one of the memory chips. The first column lists the chip's location. The second column lists the data bit of the 68000. The third column lists the pin on the 68000 to which pin 2 of the memory chip should be connected. The fourth column lists the pin on the memory buffers to which pin 14 of the memory chip should be attached.

If you discover that one of these connections is not connected properly, find and correct the broken etch, or add some small-gauge wire until the connection is restored. If all the connections are in working order and you're still having trouble, use the system diagnostics in the Macintosh ROM to identify the bad memory chips. Look at all signals connected to the suspected chip for a bad connection.

### Diagnostics

Before starting the diagnostics, you must have installed the programmer's buttons Interrupt and Reset on the left side of your Macintosh. Hold down the Interrupt button and either press the Reset button or power on your Macintosh.

A sad Macintosh icon appears with a numeric code under it. If all is working well, the code will be 0F 000D, and some small bits will cycle under the code to indicate that the Macintosh is running the memory diagnostic program. The numeric code that you will see has two parts; for example, 0F is the class code and 000D is the sub code. As shown in Table III (at left), the class code tells what part of the diagnostic program found an error, and the sub code tells what the error was. Each of the sixteen bits in the sub code identifies one of the sixteen memory chips. Table IV (at left) maps the sub code bits to their respective chip's location.

If the diagnostics discover more than one bad chip, the sub code displays multiple bits. For example, if bit 3 is bad, the diagnostics display sub code 0008. If both bit 3 and bit 10 are bad, the diagnostics display sub code 0408. If the diagnostics identify a bad chip, you'll have to replace it with a good one or find the problem on the board.

After all the diagnostics have passed, the program displays an exception code giving the current state of your MAC. You should normally see a sub code of 000D, NMI. The others are listed in Table V (at right) for your information only.

## Removing the Modification

If you need to convert your Macintosh back into a standard 128K unit, simply remove the IC assembly you added at location F3, remove the sixteen 41256 memory chips, and replace them with standard 200-ns 4164 64K memory chips. You must also connect the jumper at location E3, between pins 1 and 2.

We've completed the conversion on over 10 MACs and have had no problems with this upgrade. So good luck with your new FAT MAC.

DDJ

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 191.

## READ.ME

The 256K dynamic RAM chips that turn a Mac into a Fat Mac are very sensitive to static electricity. Please note the following advice on handling 256K RAMs, adapted from information from John Gilchrist of Microprocessors Unlimited in Beggs, Oklahoma.

To damage a true LSI device like a 256K RAM chip, you don't have to touch it. Being close to it with a high potential voltage on your body will do the job. You can, for example, generate 3000 volts by walking across a carpet in leather shoes or by peeling off a foot or so of cellophane tape, and you won't feel a thing as the tiny spark jumps to the IC, doing its hidden damage.

Following these steps should lessen the risk to the chips from static electricity.

1. If you have a choice of workspace, almost any floor covering is better (for the present purpose) than carpeting.

2. Take off your shoes. Don't take this as a personal remark, but your feet sweat enough to make it unlikely that a high static charge can build up when you are standing barefoot on a noncarpet surface. And don't wear any nylon clothing.

3. Spread out a large sheet of aluminum foil (about a three-foot length) and work on that. Wrap a corner of the foil around the Mac chassis or motherboard and poke the computer's power cord through the foil. Keep your body (e.g., one elbow) on the foil throughout the process of preparing the Mac to receive its new chips.

4. Don't handle the 256K chips until, or any more than, you have to. When you are ready for them, slide the chips out of their factory tube onto the foil. Keep your body in constant contact with the foil as you install the chips.

5. When you've finished the installation and are ready to test your work, be sure to remove the wall plug from the foil before plugging it into the wall. Otherwise you could fry more than a chip.

### Class Code

F = Exception

### Sub Code

|      |                         |
|------|-------------------------|
| 0001 | Bus error               |
| 0002 | Address error           |
| 0003 | Illegal instruction     |
| 0004 | Zero divide             |
| 0005 | Check instruction       |
| 0006 | Traps instruction       |
| 0007 | Privilege violation     |
| 0008 | Trace                   |
| 0009 | Line 1010               |
| 000A | Line 1111               |
| 000B | Other exception         |
| 000C | Nothing                 |
| 000D | NMI (normal indication) |

Table V.  
Exception Codes

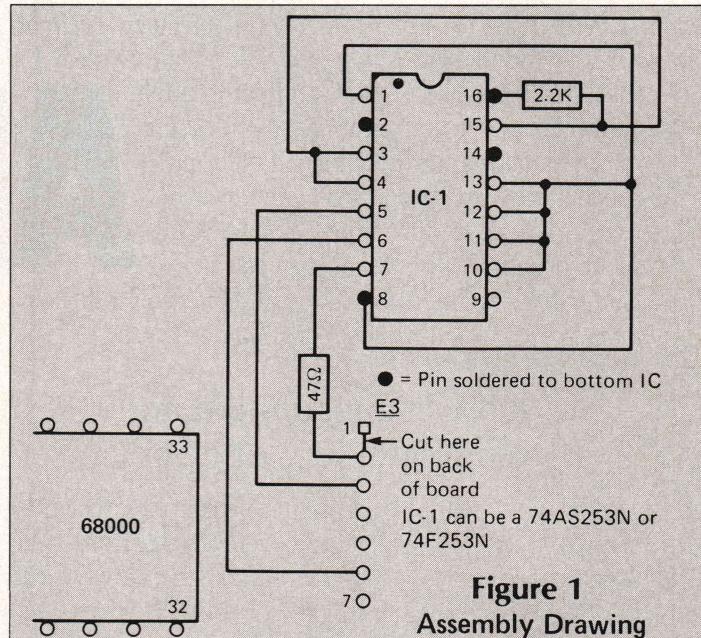


Figure 1  
Assembly Drawing

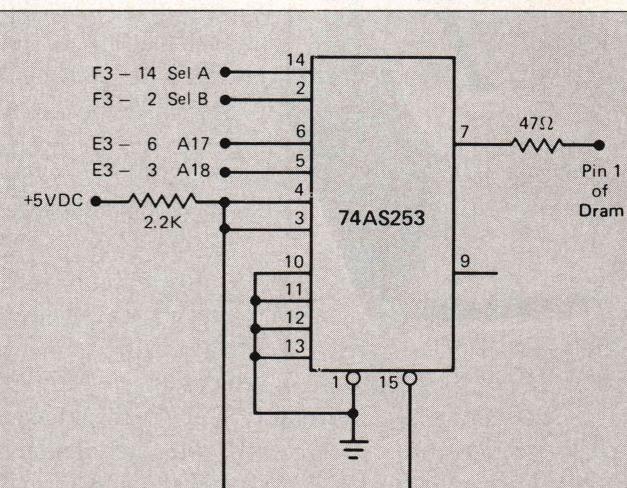


Figure 2  
Logic Drawing

## INTRODUCING Interface Technologies' Modula-2 Software Development System

The computer press is hailing Modula-2 as "the next standard in programming languages." Modula-2 combines the strengths of its popular predecessor—Pascal—with the features that made the C language appealing, like independent compilation and direct hardware control.

But until today, no company offered a Modula-2 system that made software development fast, easy and efficient.

### The fast, powerful tool for programmers

Now that breakthrough is here: Interface Technologies' Modula-2 Software Development System for the IBM® PC, XT, AT and compatible computers gives programmers the same quantum leap in productivity that spreadsheets and word processors gave to end-users. It can reduce monotonous wait time, dramatically increase speed, help eliminate thoughtless mistakes, and free you to become more creative in all your programming efforts.

### How to speed input and eliminate 30% of errors

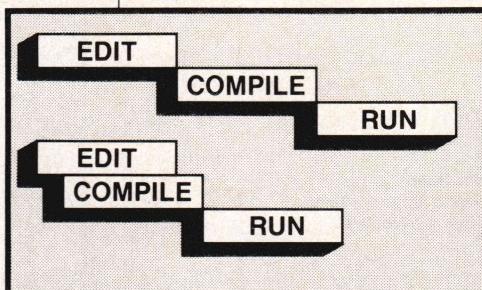
Thirty percent of programming mistakes are syntax errors and simple typos in the program structure. Our "syntax-directed" Modula-2 editor does away with these time-consuming headaches forever.

  
It also speeds input by reducing manual typing as much as 90%, letting you enter statements with a single keystroke. For example, if you type a capital "I" at the beginning of a line, the editor completes the logical "IF THEN" statement automatically, so you can concentrate on what you want to program, rather than your typing.

The editor locks out errors, finishing each statement and procedure in perfect accord with the standardized rules of Modula-2. It also indents and formats your text automatically, making programs easy to read and maintain, an important feature on big projects.

And if you leave an undefined variable or data type, the editor detects the mistake and gives you the option of on-line "help" to correct it. No other programming text editor offers you this much innovation.

### How to turn "wait time" into "work time"



*The Interface Technologies Modula-2 Software Development System saves time by compiling while you edit.*

Most of a programmer's time is spent waiting, and the biggest culprit is usually the compiler. Our compiler

# THE ANATOMY OF A

turns this wait time into work time, with a technical innovation we call "background" compilation.

With background compilation, every moment you spend writing or editing a Modula-2 program, it's automatically being compiled into object code, line by line as you work!

When you're finished editing, all that's left for the compiler to do is a quick mopping up that generates optimized native code in a single pass.

How quick is "quick"?

Thanks to background compilation and the fact that the compiler itself is so fast, Interface Technologies' compiler can turn 100 lines of typical Modula-2 program text into optimized machine code in less than five seconds.

And the Interface Technologies Modula-2 Software Development System compiler produces compact code that has execution speed superior to that produced by any other Modula-2 compiler presently available to individuals and firms involved in software development.

### How to do two things at once

Along with the syntax-directed editor and background compilation, Interface Technologies' Software

Development System gives your screen multiple windows so you can refer to one file while you edit another—simultaneously.

Concurrent editing of multiple files is particularly useful when you're doing programming work intended for separate compilation, and Interface Technologies has the only Modula-2 development system on the market that provides you this helpful benefit.

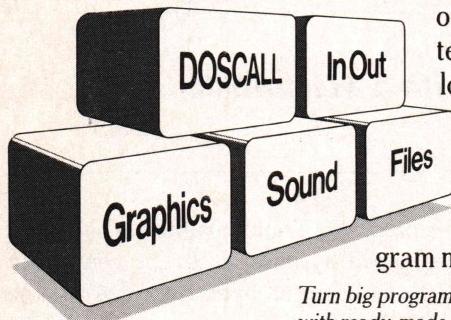


Work with multiple files faster, easier in windows.

## How preprogrammed modules speed development

One of the advantages of Modula-2 is that it lets you build large, reliable programs quickly, by linking smaller "building-block" modules.

The development system's toolkit of precompiled program modules includes the standard Modula-2 library, and adds exclusive link-and-run modules for color graphics support, sound, and direct calls to the



Turn big programs into smaller projects with ready-made modules.

## Increase productivity for \$249

Interface Technologies' Software Development System is fast, powerful and unlimited. It works so well that it's the same tool Interface Technologies is using to write business and consumer applications in Modula-2.

For \$249, you get the syntax-directed editor and compiler, linker, module library and tutorial that will have even modestly experienced programmers writing in Modula-2 in days. And you have full rights to your work; there's no license fee for programs you develop with our system.

You can use it on any IBM® PC, XT, AT or compatible with two DSDD floppy drives and 320K RAM.

You get a thoroughly indexed, comprehensive user's manual and free telephone support from Interface Technologies.

But the most important thing you get is the future, and the programming language of the future is Modula-2.

For more information, or to order the Modula-2 Software Development System, call 1-800-922-9049 today. In Texas, call (713) 523-8422.

To order by mail, or to request further information, fill out and mail the coupon below.



IBM is a registered trademark of International Business Machines Corporation.

# BREAKTHROUGH

operating system. Plus you get low-cost updates from Interface Technologies' growing library of program modules.

Turn big programs into smaller projects with ready-made modules.

NAME \_\_\_\_\_ PHONE \_\_\_\_\_

COMPANY or SCHOOL \_\_\_\_\_

MAILING ADDRESS \_\_\_\_\_

CITY \_\_\_\_\_ STATE \_\_\_\_\_ ZIP \_\_\_\_\_

PLEASE SEND ME \_\_\_\_\_ copies @ \$249 each.

MAIL REQUESTS TO:

INTERFACE TECHNOLOGIES CORPORATION

3336 Richmond, Houston, Texas 77098

ITC will accept personal checks, money orders or the following credit cards:

Visa/American Express/MasterCard

Credit Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Interbank number {MasterCard}: \_\_\_\_\_

Your Signature \_\_\_\_\_

Texas residents, add 6.125% Sales Tax.

**INTERFACE TECHNOLOGIES**



**MODULA-2 SOFTWARE  
DEVELOPMENT SYSTEM**

# QuickDraw Meets ImageWriter

by Thom Mayer

Apple's new crop of computers, the Macintosh and the Lisa, makes extensive use of bit-mapped graphics. Less visible than the graphics hardware, but just as necessary, is the well thought-out, well crafted, integrated collection of graphics software primitives called QuickDraw. These routines make it a simple matter to put graphics into programs. Because of QuickDraw, programmers have an easier job, users get better software, and Apple—who paid for its development—racks up sales. Unfortunately, the Lisa QuickDraw routines lack any support for the Apple ImageWriter dot matrix printer; hence, this article. The ImageWriter exhibits the quality features we have begun to associate with Apple hardware (e.g., well-crafted built-in software primitives and complete concise documentation). I would recommend the ImageWriter to anyone in the market for a printer.

look at the routine IMAGE\_PRINT and discuss some of the considerations that influenced its design.

QuickDraw stores an image in a continuous piece of memory with each bit corresponding to one pixel: a one for black, a zero for white. For example, the Lisa screen, which measures 720 × 364 dots, requires 262,080 bits or 32K bytes. The exact relationship between the data in memory and the pixels on the screen depends on the values stored in the bitMap. A bitMap is a data structure specified in Pascal as follows:

```
BitMap = RECORD  
  baseAddr : QDPtr;  
  rowBytes : Integer;  
  bounds : Rect;  
END;
```

The field baseAddr is a pointer (memory address) to the beginning of the block of memory where the image is

## *Printing sideways with a Macintosh.*

Figures 1 and 2 (page 27) were drawn on a Lisa by QuickDraw and were printed on an ImageWriter dot matrix printer with the enclosed routine. Listing One (page 31) gives a Pascal unit containing the function IMAGE\_PRINT, and Listing Two (page 35) demonstrates its use. In this article we will briefly review the way QuickDraw stores a graphics image and the way the ImageWriter prints graphics. Then we will take a cursory

stored. We should imagine this memory grouped by lines, each line containing the number of bytes specified by the field rowBytes. Two dots on the screen that have the same vertical coordinate and differ in the horizontal coordinate by one are neighboring bits in memory. Two dots on a QuickDraw image that have the same horizontal coordinate and differ in the vertical coordinate by one are separated by exactly rowBytes bytes in memory (Figure 3, page 27).

The ImageWriter, like all dot matrix printers, forms characters and graphics by printing a pattern of dots. The printer head is composed of a vertical array of nine dot-forming devices spaced 1/72 inch apart; at one stroke, the Im-

*Thom Mayer, Tigre Designs, 3006 Lafayette, Austin, TX 78722.*

ageWriter produces a pattern of nine vertical pixels. Normal characters are composed of seven such patterns, with a blank eighth pattern forming the space between the characters. The horizontal spacing between successive patterns can be set by sending the printer control codes: A spacing of 1/72 inch yields nine characters per inch, a spacing of 1/136 inch yields 17 characters per inch, and so on (Figure 4a, page 28).

When the ImageWriter is in graphics mode, each byte received produces an eight-pixel pattern determined by the binary representation of the byte (recall 1 byte = 8 bits). The top pixel corresponds to the least significant bit, and the bottom pixel corresponds to the most significant bit. A dot is printed if there is a one in the corresponding bit; no dot is printed if there is a zero (Figure 4b, page 28).

Because a byte in QuickDraw corresponds to eight horizontal pixels on the screen, and a byte to the ImageWriter prints eight pixels aligned vertically, the image will come out sideways on the paper. Each printed line of graphics corresponds to a vertical strip on the screen, eight pixels wide.

The pixels on the Lisa screen are not square: they are 1.5 times as high as they are wide. On the printer, the vertical spacing between dots is fixed by the printer head to 1/72 inch, so to match the Lisa screen aspect ratio we need a horizontal dot length 1/48 of an inch, which we make out of two dots on 1/96-inch spacing. At this spacing, an image the size of the Lisa screen nearly fills an  $8 \times 10$  sheet of paper.

The printer algorithm naturally breaks into several steps, shown here in the simple form of the main routine:

```
function image_print ({parameters}) : boolean;
begin
  size_of_image;
  init_printer;
  init_buffer;
  for line_number := 1
    to (row_width) do
      begin
        buffer_a_line
          (line_number);
        spew_buffer;
      end;
  close_printer;
  image_print := true;
```

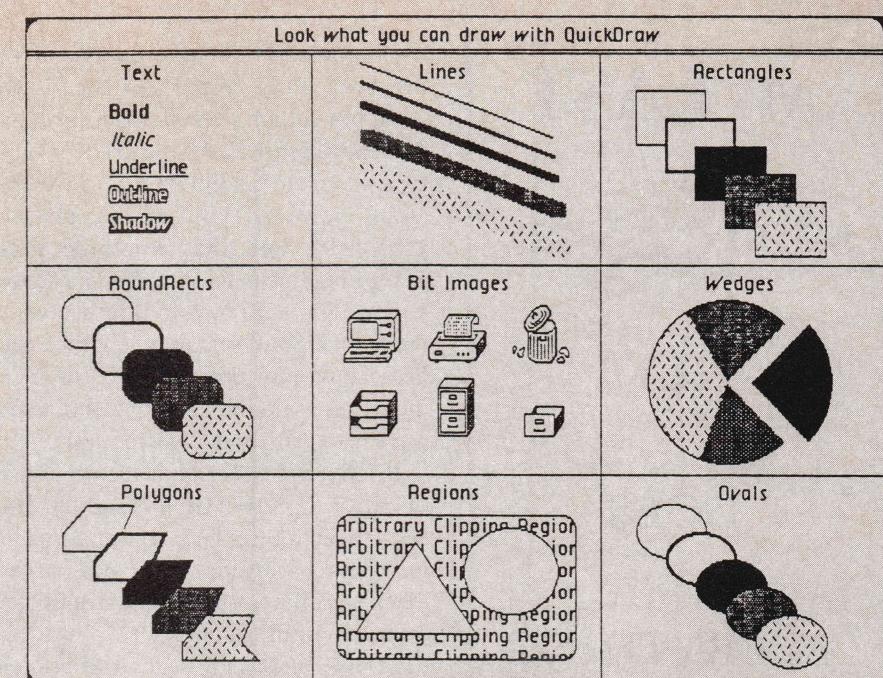


Figure 1.

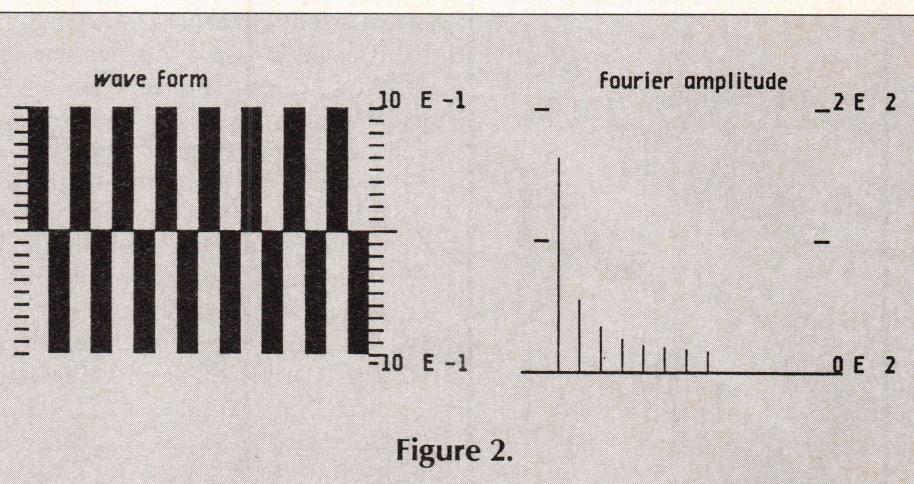
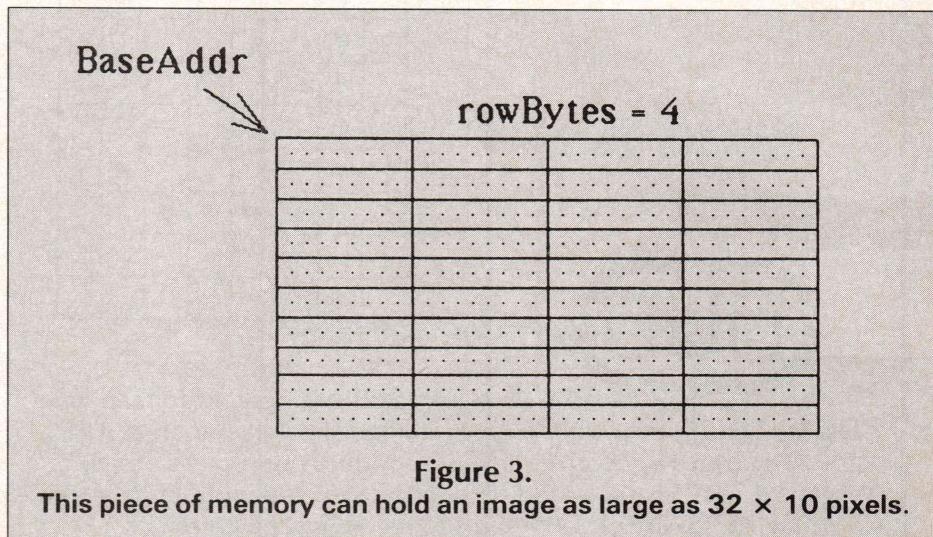
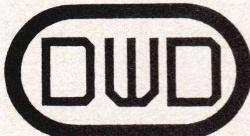


Figure 2.



# AT LAST S-100 ↔ 488 THAT DOES EVERYTHING YOU WANT IT TO DO



D&W DIGITAL, INC.  
20655 Hathaway Avenue  
Hayward, California 94541  
(415) 887-5711

end;

This algorithm did not spring out of my head fully formed in top down style, but rather I developed it through successive refinement of a crude prototype. At some point in the development, however, you must translate your work into the structured programming form. This process may give you insights into how to improve your work, and it will certainly lengthen the lifespan of your programs by simplifying future modifications. Good software is too costly to be disposable.

Briefly let's go through each routine: (1) SIZE\_OF\_IMAGE looks at the bit-Map to see where the image is stored in memory (baseAddr), how it is organized (rowBytes), and what size image we are to print (bounds).

(2) INIT\_PRINTER sends the printer the initialization codes. To change from the default settings, which are appropriate for text, to the proper settings for graphics takes 16 bytes of data, so most of this first block is NULS.

(3) The FOR loop prints one line of graphics per pass. BUFFER\_A\_LINE gathers the bytes needed to print the line and stores them in the buffer. SPEW\_BUFFER sends the buffer to the printer and listens for confirmation from the printer. At the beginning and end of the buffer are control codes that are initialized by INIT\_BUFFER.

(4) CLOSE\_PRINTER close files, switches the printer back to default

settings, and form-feeds the paper. If the routine gets this far, it has detected no problems and returns the value true.

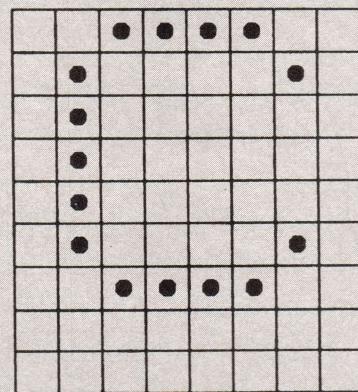
There we have it, the means to print a QuickDraw image on the Image-Writer dot matrix printer. Now we will look more closely at the program design, so have your listing handy.

IMAGE\_PRINT is set up as a Boolean-valued function that returns false only if the routine couldn't print the image. The usual cause for failure is the printer being off, which is detected when the blockwrite routine says it was unable to complete the transfer of blocks. All failures are channeled through the procedure NO\_GOOD :

```
procedure no_good;
begin
  image_print:=false;
  exit(image_print);
end;
```

The not well-known but extremely handy procedure exit satisfies those programming urges that in other environments might be handled by a GOTO statement. It is the calling program's responsibility to respond appropriately to failures. The following code fragment, which uses the system call PAabortFlag, gives the user the ability to recover or abort:

```
while not ( PAabortFlag or
  IMAGE_PRINT
  (grafptr,printerPort) )
begin
  writeln (' fix printer or press
```

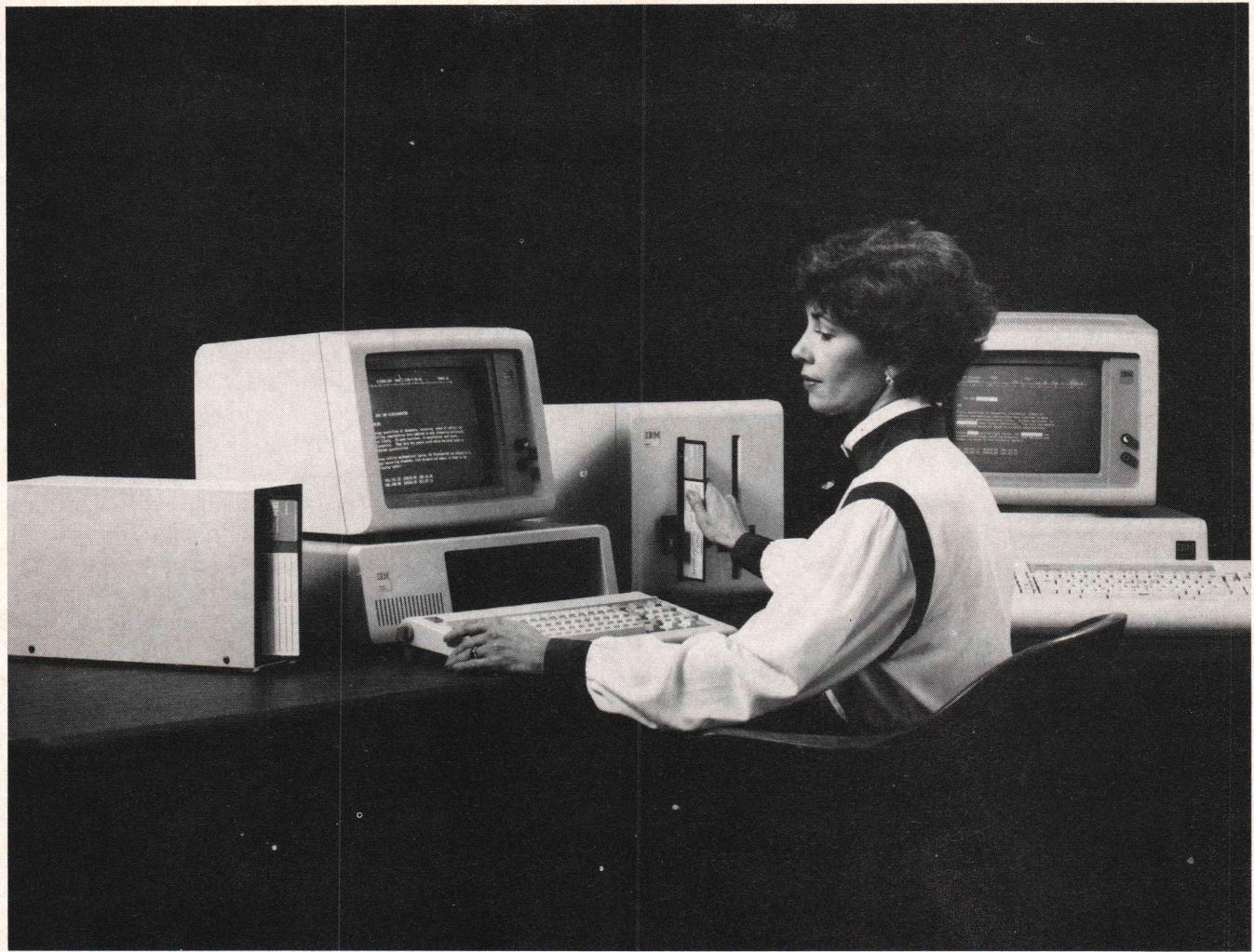


**Figure 4a.**  
The printer head produces nine vertical pixels per shot. The standard characters are constructed from eight such vertical patterns.



**Figure 4b.**  
In graphics mode, each byte produces a vertical line of eight pixels. This pattern was produced by the byte 01101011 binary (i.e., 107 decimal).

# Change your diskette to fit the IBM PC



## THE FILE CONNECTION 8" DISKETTE SYSTEM FOR THE IBM PC

Our "FILE CONNECTION" programs provide 8" diskette file exchange between the IBM PC and most Micro-Mini-Main Frame computer systems.

Our "WORD CONNECTION" programs provide 8" diskette text document exchange between the IBM PC and many word processing systems.

Our "DISPLAYWRITER CONNECTION" programs transform documents from Textpack, Wordstar, Multimate, etc. to the DisplayWrite 2 format.

In addition to our hardware and program products, we also provide a conversion service for customer supplied diskettes. Please contact us for information about the hundreds of 5 $\frac{1}{4}$ " and 8" diskette formats and systems which we currently support.

**FLAGSTAFF ENGINEERING / P.O. Box 1970 / Flagstaff, AZ 86002**  
**Telephone 602-774-5188 / Telex 705609 FLAG-ENG-UD**

# About ZCPR3 and Z3-DOT-COM

Flexibility is the key to ZCPR3 power. Productivity results from optimum organization of operator and machine resources. You are free to create a thinking and working environment that you choose!

ZCPR3 permits quick computer system re-organization for varying tasks, controls your application programs from integrated, easily produced and changed menus. Generation of aliases permit many commands and keystrokes to be converted to a few. Aliases may be used from within menus. One alias may use another. Control is near absolute using supplied utilities. Menu generation determines how computer is used, simplifying and speeding operations. Single from-menu keystrokes activate complex series of commands.

Shells, multiple commands per line, named directories, file search paths, if-then-goto conditional processing, screen oriented utilities — all major features.

Utilities provided permit file and disk management, easy coordination of many application programs from chained menus with full security and password protection. Online and built-in HELP assists understanding details of each command. ZCPR3, the definitive 8-bit CPM-80 compatible operating system, is a hard worker — one you use, learn from, grow and live with!

ZCPR3 is available in two versions: 1) manual-install system using CP/M MOVCPM, SYSGEN, DDT, and MAC; and 2) auto-install Z3-Dot-COM version. Z3-Dot-COM installation procedure is detailed in eleven (11) lines at the bottom of a Command Reference card. The manual-install version is for two-year-or-over computer users and programmers; the auto-install is ideal for CP/M beginners.

## 1. Z3-DOT-COM

Auto-Install, load and go, complete full-up ready-to-run system on 4 disks ..... \$149.00

## 2. ZCPR3 Core and Utilities

Manual-Install, source to everything on 10 disks with installation procedure ..... \$128.00

## 3. ZCPR3: The Manual

Lavish, typeset, over 300 pages ..... \$19.95

## 4. DISCAT

Fancy menu-driven disk catalog system ..... \$49.00

A fortnighter newsletter, 24-hour BBS and RCP/M System keep Z3 users informed of microcomputer happenings. Order now! State CP/M disk format desired; add \$3.00 shipping & handling; Californians please add 6.5% sales tax. Visa/MC, check, money or purchase order accepted.

(Trademark: CP/M, Digital Research)



**Echelon, Inc.**

101 First Street • Los Altos, California 94022 • 415/948-3820

```
abort key');
writeln (' press RETURN to
continue ');
readln ( );
end;
```

Filling the buffer with the bytes needed to print one line is like buying groceries at the supermarket. BUFFER\_LINE loads a byte, skips row\_bytes bytes, loads a byte, and so on; clearly we need random access to memory by the byte. This calls for a packed array of char. But we don't want to make a new array: we want to use the memory that QuickDraw has already set up. So rather than declaring an array variable, we declare a pointer variable, CH, of type pointer to packed array of char. Now we are all set. Assign CH the value baseAddr (i.e., the beginning of the image memory), and an expression like CH^[n] gives access to the nth byte of the image memory. The memory has not been changed—only the way the program views it.

Pascal is a strongly typed language, and most compilers check the type of a pointer before allowing assignment. Just as you can't mix apples with oranges, you can't mix pointers to apples with pointers to oranges. Lisa Pascal, however, includes a wonderful extension, the unary operator @, which returns the pointer to the operand (just like the & operator in C). For example, if Z is an array, then @Z is a pointer that points to Z. Simple, yes? The @ operator has another important feature: the pointer resulting from an @ operation, like the nil pointer, is assignment-compatible with any other type pointer. To make CH point to the same place as bitAddr, use the assignment CH := @bitAddr^. I call this de-typing a pointer. It is like a CAST operation in C.

Between the routine IMAGE\_PRINT and the printer sits the device driver routine that actually transfers the bytes. A feature of the Lisa port driver is the automatic insertion of a line feed after each carriage return, an option that can be enabled and disabled with system calls. The automatic line feed may be handy if you have a vintage 1960 printer, but in a graphics setting, adding the byte 00001010 behind every occurrence of the byte

0001101 is a bug. To track down this bug, without the 20-20 vision of hindsight, required a long session with the debugger. Among other capabilities, the debugger allows you to halt the program and to go in and look at the memory—even change it if you wish—and then resume operation of your program. My search went as follows.

First I went in and looked at the memory where the screen image was stored to make sure that it looked as I had expected. Then I looked at the memory corresponding to my buffer to make sure it was correct. Upon closely inspecting the buffer for the line that was giving me trouble, I found out that the bug happened only when the byte 0D was in the buffer. With the debugger I could change the occurrences of 0D to some other byte, or vice versa, and ascertain that 0D was the guilty party. 0D is the code for carriage return, and it was instantly clear to me that somebody was adding an LF after every carriage return. But who?

I spent a long time thinking the

printer was the culprit but finally convinced myself otherwise and uncovered the automatic line feed feature of the port driver. The system calls necessary to reset this appear in the procedure AUTO\_LF, which can enable or disable the automatic line feed on either serial port. Unfortunately, the routine now needs to know which port the printer is connected to. I was tempted to circumvent this need by temporarily resetting both port drivers, but this would be unwise on a multitasking machine like the Lisa because some background task might be using the other port; for example, a modem connected to the other serial port might be in use by a BBS running in the background.

In the interest of simplicity, this version of IMAGE\_PRINT makes no attempt at treating large blank spaces intelligently, an addition that can improve speed substantially. As presented, IMAGE\_PRINT requires 188 seconds to print a  $720 \times 364$  picture. Also in interest of simplicity, the width of the printed image is rounded up to

the nearest multiple of eight, i.e., on a byte boundary. For those desiring sample QuickDraw programs and a version of IMAGE\_PRINT that does not make the above two simplifications, mail me a \$25 check, and I will send you a 3.5-inch Lisa format disk with two versions of IMAGE\_PRINT, other graphics utilities, and complete programs that use QuickDraw and the utilities. Source and compiled code is included. You are welcome to distribute the code for noncommercial purposes as long as you do not remove my copyright notice. Please identify any modifications you make, and I encourage you to share any successes you have. We are the neurons of our cultural brain.

DDJ

#### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 192.

## QuickDraw Meets ImageWriter

(Text begins on page 26)

### Listing One

```
{  
{$R-}  
unit GPHU; {graphics utilities }  
{ COPYRIGHT T.MAYER 1984 -- DISTRIBUTION FOR NON-COMERCIAL PURPOSES ALLOWED }  
interface  
  uses  
{$U-}  
    {$U QD/QuickDraw } QuickDraw,  
    {$U QD/QDSupport } QDSupport,  
    {$U syscall} syscall;  
{$U+}  
  
  function image_print(tp:grafptr;printerPort:integer) : boolean;  
  
implementation  
  
function image_print;  
type cars = packed array[0..maxint] of char;  
var  
  printer:text;p:file;  
  row_bytes,bytes_to_print,height,line_number,index,i,buff_top:integer;  
  ch:^cars;  
  buff : packed array[0 .. 1023] of char;  
  
procedure AutoLinefeed(whichPort:integer;enabled:boolean);  
{ whichPort=0 -> portA, whichPort=1 -> portB }  
var Ecode:integer;path:pathname;Cparm:Dctype;  
begin  
  if (whichport mod 2=0) then path:='RS232A' else path:='RS232B';  
  CParm.dcVersion:=2; CParm.dcCode:=17;  
  if enabled then CParm.dcData[0]:=1 else CParm.dcData[0]:=0;  
  DEVICE_CONTROL(Ecode,path,Cparm);  
end;
```

(Continued on next page)

# QuickDraw Meets ImageWriter

(Listing Continued, text begins on page 26)

## Listing One

```
procedure no_good;
begin
    image_print:=false;
    exit(image_print);
end;

procedure init_printer;
var i:integer;
begin
{ first we disable auto line feed in RS232 device driver }
    AutoLineFeed(PrinterPort,false);

{ the printer file is to be accessed via blockwrite. }
    reset(p,'-printer');

{ send one block, mostly nuls, with initialization codes :}

{this tells printer to recognize 8th bit - neccessary for graphics }
    buff[0]:=chr(27);
    buff[1]:=chr(90);
    buff[2]:=chr(0);
    buff[3]:=chr(50);
{make sure printer adds no LF after CR }
    buff[4]:=chr(27);
    buff[5]:=chr(90);
    buff[6]:=chr(128);
    buff[7]:=chr(0);

{ this sets horizontal spacing to 96 dots per inch }
    buff[8]:=chr(27);
    buff[9]:='E';

{ this sets vertical line feed to 1/9 inch, i.e. height of 8 dots }
    buff[10]:=chr(27);
    buff[11]:='T';
    buff[12]:='1';
    buff[13]:='6';

{this sets printer head motion left to right only. yields better quality output }
    buff[14]:=chr(27);
    buff[15]:='>';

    for i:=16 to 511 do buff[i]:=chr(0); {most of block is NULs }

{ now actually send the initialization block }
    if (blockwrite(p,buff,1)<>1) then no_good;

end;

procedure init_buffer;
var i:integer;
begin
{ these first six bytes say "here comes 728 graphics codes" }
    buff[0]:=chr(27);
    buff[1]:='G';
    buff[2]:='0';
    buff[3]:='7';
    buff[4]:='2';
    buff[5]:='8';

    for i:=6 to 1023 do buff[i]:=chr(0); {blank if not otherwise reset }

    buff[734]:=chr(13);buff[735]:=chr(10); { CR LF ends each line}

    buff_top:=6;
```

(Continued on page 34)

*Announcing a*

# TOTAL PARSER GENERATOR

**<GOAL> :: = <RAPID> <COMPILER> <DESIGN>**

## SLICE YOUR COMPILER DEVELOPMENT TIME

An LR(1) parser generator and several sample compilers,  
all in Pascal for your microcomputer.

- Generates parser, lexical analyzer and skeleton semantics
- Universal, error recovery system
- Adaptable to other languages and computers
- Interactive debugging support
- Thorough documentation
- TURBO PASCAL™ INCLUDED FREE OF CHARGE
- Includes mini-Pascal compiler, assembler, simulator in SOURCE

### SPECIAL INTRODUCTORY OFFER \$ 995

QPARSER™ runs on IBM PC/DOS in Turbo Pascal. Parser generator in object form; all else in source. QPARSER takes a grammar and generates a correct, complete, high-performance compiler with skeleton semantics in Pascal source. Easy to add full semantics for YOUR application. Excellent for industrial and academic use. An accompanying textbook (SRA publishers) available in 1985. Training can be arranged. Demo disk available for \$50.

Educational and quantity discounts available. Check, money order, Mastercard, Visa. California residents add 6.5% sales tax.

WRITE OR CALL FOR FREE BROCHURE.

Technical details: call 408/255-5574. Immediate delivery. CALL TODAY!



1164 Hyde Ave., San Jose, CA 95129

**TOLL FREE: 800-538-9787**

(California residents call 408/255-5574)

TM Turbo Pascal is a registered trademark of Borland International.

Circle no. 76 on reader service card.

# QuickDraw Meets ImageWriter

(Listing Continued, text begins on page 26)

## Listing One

```
end;

procedure enbuff(a:char);
begin
    if buff_top=1022 then exit(enbuff);
    buff[buff_top]:=a;buff[buff_top+1]:=a;
    buff_top:=buff_top+2;
end;

procedure spew_buffer;
const nblocks=2;
begin
    if (blockwrite(p,buff,nblocks)<>nblocks) then no_good;
    buff_top:=6;
end;

procedure close_printer;
var printer:text;
begin
    close(p);
    rewrite(printer,'-printer');
    write(printer,chr(27),'c');           {return printer to default set up}
    write(printer,chr(12));               {form feed}
    close(printer);
    AutoLineFeed(PrinterPort,true);     {enable port driver autoLF function }
end;

procedure buffer_a_line(line_number:integer);
var i:integer;
begin
    index:=row_bytes - line_number;
    for i:=1 to height do   {buffer a line}
        begin
        enbuff(ch^[index]);
        index:=index+row_bytes;
        end;
end;

procedure size_of_image;
var width : integer;
begin
    with tp^.portBits do
    begin
        row_bytes:=rowbytes;
        height:=bounds.bottom-bounds.top;
        width := bounds.right - bounds.left;
        ch:=#baseAddr^;
    end;
    bytes_to_print:=width div 8;
    if width mod 8 <>0 then bytes_to_print:=bytes_to_print+1;
end;

begin
    size_of_image;
    init_printer;
    init_buffer;
    for Line_number:=1 to (bytes_to_print) do
        begin
            buffer_a_line(line_number);
            spew_buffer;
        end;
    close_printer;
    image_print:=true;
end;  { of image_print }
end.  { of implementation }
```

End Listing One

## **Listing Two**

```

program example; { this sample program creates ILLUSTRATION 1 }
uses
  {$U QD/QuickDraw} QuickDraw,
  {$U QD/QDSupport} QDSupport,
  {$U syscall} syscall,
  {$U gphu} gphu,           { graphics utilities - here we use :
                                init_graphics
                                bar_plot
                                image_print }

  {$U mathlib}      mathlib; { math library - here we use :
                                square_wave
                                famp ( i.e. FourierAMplitudes ) }

var
  wave : array[1 .. 256] of real;
  i,j:integer;
begin
  init_graphics;   erase_screen;

{ make a square wave and graph it : }
square_wave(@wave,256,16);
bar_graph ( 256, @wave[1], 10,40, 256,128, 0.0, 1.0, true,false, 'wave form','');
{ fourier transform it and graph that : }
famp(256,@wave);
bar_graph ( 129, @wave, 400,40, 200,128, 0.0, 1.0, true,false, 'fourier amplitude');

{ print the whole screen }
if not image_print ( thePort, 0 ) then writeln('turn on the printer');

end.

```

End Listings

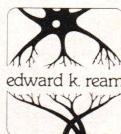
## **C Source Code**

### **RED**

#### **Full Screen Text Editor**

**IBM PC, Kaypro, CP/M 80 and CP/M 68K systems.**

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
- RED is easy to use for writers or programmers. RED's commands are in plain English.
- RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.



Call or write today for  
for more information:

Edward K. Ream  
1850 Summit Avenue  
Madison, WI 53705  
(608) 231-2925

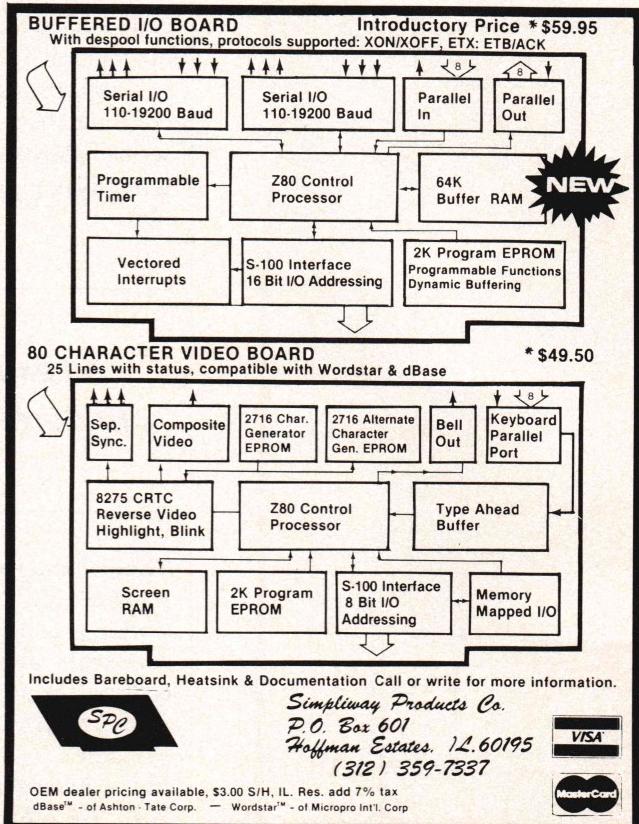
To order:

Either the BDS C compiler or the Aztec CII compiler is required for CP/M80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5 1/4 inch disk).

Send a check or money order for \$95 (\$105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.



Circle no. 87 on reader service card.

# Archiving Files with CP/M-80 and CP/M-86

by Ian E. Ashdown

If you use CP/M-80 versions 2.x or CP/M-86, you know the problem well: sitting there at two in the morning trying to remember which files you worked on so you can copy them to a backup disk. If you have a hard disk in your system, the problem can be an acute pain. Which of several hundred files did you update or otherwise modify during your marathon programming session?

The ideal solution would be to have a utility program that somehow determines which files have been changed on a disk and automatically copies them to a backup disk. This procedure, known as "incremental backup," is superior by far to the usual methods of either relying on your memory (at two in the morning?) or copying the entire disk.

Although Digital Research's documentation for their CP/M-80 and CP/M-86 operating systems gives no indication that a file is marked in any manner when it is written to or renamed, you nevertheless can implement an incremental backup utility exactly as described above—the ideal solution. BU is one example of such an implementation.

have been written to give even the novice C programmer a clear understanding of what is going on each step of the way. What follows is a general description that covers the salient features of BU from a user's viewpoint.

The *CP/M-80 Version 2 Interface Guide*'s description of BDOS Service 30 (Set File Attributes) states that the file attribute bit t3-prime is "reserved for future system expansion." However, if you use a disk utility to set t3-prime to true in the file's directory entry, you will find that the BDOS resets it to false (zero) whenever the directory entry is changed. Since this means that the file has been opened, written to, and closed (or else renamed) by the BDOS, t3-prime is apparently an undocumented attribute bit that indicates when a file has been updated.

This behavior is not an unreliable artifact of some other process; DRI added a very similar feature, called the "Archive" attribute, to its multiuser MP/M 2 operating system. The version of PIP.COM supplied with MP/M 2 features an "A" option, which causes PIP to copy only those files that have their Archive bits set false. After copying each

## An expedition into the jungle of Undocumented Features comes home with an orphan attribute bit.

### Theory and Practice

For a detailed explanation of the inner workings of BU, you should read the comments accompanying the source code in the Listing (page 39). These

Ian Ashdown, byHeart Software, 2 – 2016 West First Avenue, Vancouver, B.C. V6J 1G8.

file, PIP sets the bit true. It seems logical, then, that in writing CP/M-80 versions 2.x and CP/M-86, DRI intended to rewrite its version of PIP to include an "A" option but for whatever reason never got around to doing so. This leaves the user to come up with a utility that takes advantage of this orphaned attribute.

A variety of such utilities are avail-

able, including one in the public domain and at least two commercial products. BU's advantage is that it is written in C, thus presenting you with the opportunity to customize it to your particular needs. The source code has been profusely commented for precisely this reason.

BU accepts command lines of the following form:

BU x[:afn] y [-AFHQSn]

where x = drive name of disk to be backed up,  
y = drive name of backup disk,

and the optional arguments are:

- A All files, regardless of status
- F Fast copy (without verification)
- H Hard disk (files may be split)
- Q Query each file before backup
- S System attribute copied to backup
- n Backup USER n files only (0 - 31)
- afn Any legal CP/M ambiguous fileref (can only be used with -n option)

If the above is a bit confusing, some examples may help in explaining the various options:

**BU a b** – Copy updated files in all user areas from drive A: to drive B:.

**BU c a -f** – Copy updated files in all user areas from drive C: to drive A: without verification of copied files.

**BU a:file.typ m -5** – Copy file A:FILE.TYP (user area 5) to same user area on drive M:.

**BU a:file\*.t? c -0q** – Copy any files in user area 0 matching ambiguous file reference A:FILE\*.T? to the same user area on drive C:. The operator is queried before each file is copied. Answering y or Y for "Yes" results in the file being archived; anything else results in the file being skipped.

**BU b a -ah** – Copy all files from all user areas from drive B: to drive A:. If BU runs out of backup disk space while copying a file, the file will be split across two disks.

**BU a b -a -s** – Copy all files from all user areas from drive A: to drive B:. Those files with the System attribute set are copied as System

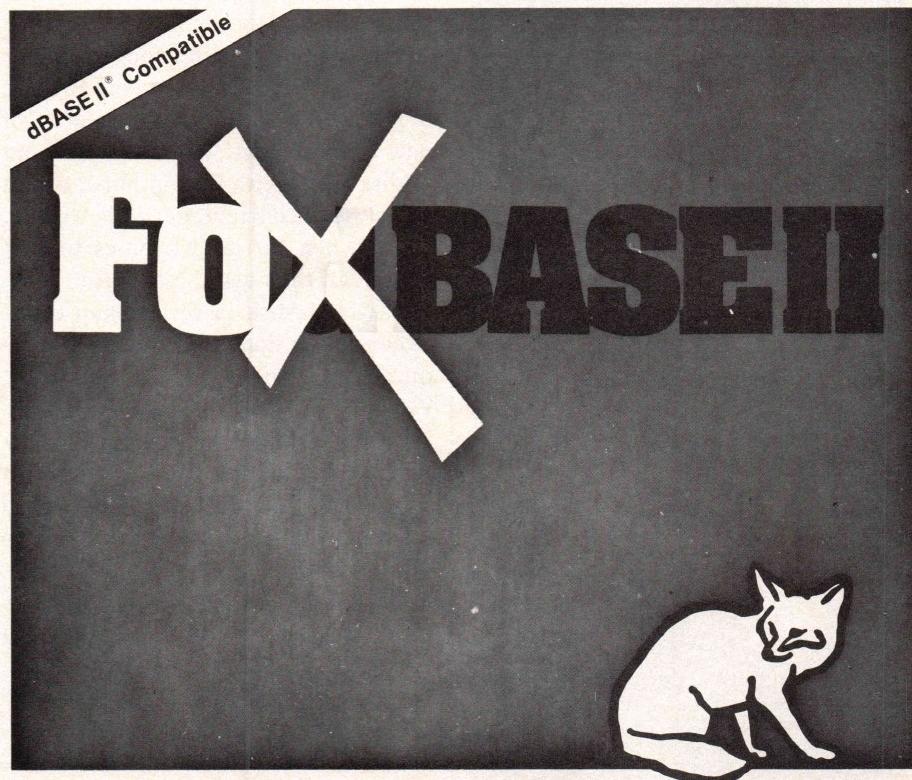
files to drive B:. (Note that the dash options can be separated.)

A fair amount of the code involved in BU has to do with defensive programming: always assume that the user will make a mistake. The command line is validated as thoroughly as possible. Any errors detected are displayed with an appropriate message, along with the previously listed explanation of what command lines are valid.

Once in operation, assuming no options have been selected or ambiguous file references specified, the program

will scan the directory of the disk in drive x, note which files have been changed since the last time BU was run on that disk, then copy only those files to the disk in drive y. Existing files with the same fileref and user number on the backup disk are automatically erased.

Because the new files are backup copies, each one is read after it is written and verified character by character with the original file. All available memory is used to buffer the data for disk read and write so that BU can copy and verify as quickly as possible. Once the new file has been fully verified, its



## dBASE II® outfoxed!

Who says the most popular database management system is the best?

Introducing FoxBASE II™! the new relational database management system that's dBASE II source compatible. It does everything dBASE II does ... plus a whole lot more.

- Runs 3 to 5 times faster
- Sorts up to 20 times faster
- Permits up to 48 fields per record ... 50% more than dBASE II
- Supports full type-ahead
- Compiles program sources into compact object code
- Comes with a sophisticated online manual and HELP facility
- Has twice as many memory variables

What's more, it costs less. **MS-DOS \$395. AOS/VS \$995.**

FoxBASE II is available now for: IBM-PC, IBM-PC/XT, COMPAQ & IBM compatibles, TI Professional, DG Desktop, DG MV Series, and many others. Call or write today for more information.

Developed by

**DACOR**

COMPUTER SYSTEMS 13330 Bishop Road, P.O. Box 269, Bowling Green, OH 43402 / 419-354-3981 / TWX 810-499-2989

dBASE II is a registered trademark of Ashton-Tate  
FoxBASE II is a trademark of Fox Software Inc.

**FOXBASE II**  
from FOX SOFTWARE INC.

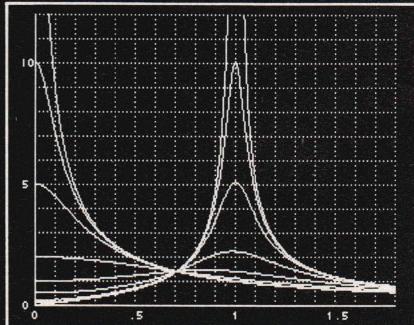


Circle no. 40 on reader service card.

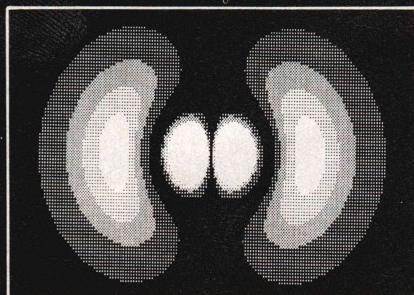
# isys FORTH

for the Apple® ] [

**Fixed point speed** can rival that of floating point hardware. But the details have been a well kept secret—until now. The following graphs were generated by fixed point examples from the ISYS FORTH manual.



Parallel Resonance with Damping  
BASIC 213 sec ISYS FORTH 27 sec



Hydrogen 3p Orbital Cross-section  
BASIC 492 sec ISYS FORTH 39 sec

- Fast native code compilation. Sieve benchmark: 33 sec
- Floating Point—single precision with transcendental functions
- Graphics—turtle & cartesian with 70-column character set
- Double Precision including D\*/D
- DOS 3.3 Files read & written
- FORTH-83 with standard blocks
- Full-Screen Editor
- Formatter for word processing
- Macro Assembler
- Price: \$99, no extra charges

## ILLYES SYSTEMS

PO Box 2516, Sta A  
Champaign, IL 61820

Technical Information:  
217/359-6039, mornings

For any Apple ] [ model, 48K or larger.  
Apple is a registered trademark of Apple Computer.

file attributes are set to "directory" (DIR) and "read-only" (R/O) to ensure that it can be displayed in a directory listing of the backup disk and that it cannot be accidentally erased.

If the combined size of the files to be backed up exceeds the available space on the backup disk, BU will take one of two actions, depending on whether or not the -H option has been selected. In the default mode, BU will stop when it runs out of disk space and erase the current, partially written, backup file. It will then ask the operator to insert a fresh disk in the backup drive. When this is done, BU will begin to copy files to the new disk.

The -H option is intended primarily for use with hard disks, where the size of the files may exceed the capacity of new backup disks. When BU runs out of disk space with this option active, it will close the current, partially written, backup file, set its attributes to DIR and R/O, then ask the operator to insert a fresh disk in the backup drive. When this is done, BU will open a sequentially numbered fileref (e.g., FILE.TYP would become FILE-01.TYP) and continue writing the current file to this new backup file from where it left off. The file in effect will be split across two or more backup disks, with no wasted disk space.

Reassembling these split files is quite simple. In principle, you need only open the first file for write access, use lseek( ) to find its end, open the second file for read access, then append it to the first file. The C code required to do this is left to the reader as an exercise. Alternatively, you can always use the concatenation feature of DRI's PIP.COM utility. The command line would be:

PIP rebuilt.fil = first.fil,second.fil

The disadvantage of this approach is that all three files must be on-line at the same time, which in a two-drive system means shuffling one of them to the destination disk before you can concatenate and rebuild your original file.

What BU does not address is archival file maintenance procedures. If BU is used with a dual floppy drive system and every working disk has its own backup copy, there is no problem. At the end of your programming session,

simply insert the backup disk in the second drive, type "BU x y," and every changed file is automatically updated. The backup disk becomes a duplicate of the working disk (although you do have to manually erase files that were deleted from the working disk).

The problem arises when you want to maintain backup copies of hard disk files. Depending on whether or not the -H option is used, files will be on different disks and possibly split across disks after you run BU. When the files later are changed again, it may happen that BU will archive them on different disks than the backup copies. BU automatically erases existing files with the same fileref and user number on the backup disk before copying a file, but it can't do that when the file is on another disk. This leaves the responsibility of deleting (or otherwise archiving) obsolete versions of files to the user.

It is a very simple matter to extend BU so that a disk identifier file is added to each backup disk. This would be especially useful when using multiple floppies to archive hard disk files; the identifier files could be numbered sequentially. However, because disk naming and storage procedures are very much a matter of individual taste, this has been left to the reader to implement.

## Etymological Nuisances

BU will mishandle random access files that have been created with unwritten records or unallocated blocks or extents. Because it uses the BDOS sequential read service to access the files, BU will stop reading random access files at the first unallocated block or extent. Happily, very few programs behave in such an unfriendly manner. (This is perhaps because most file copy utilities will balk at such files as well.)

Speaking of file copy utilities, a few are available that, under certain circumstances, will write a file without resetting its Archive attribute. One of these, oddly enough, is DRI's own PIP.COM. If a file on a disk has its Archive and Read-Only attribute bits set true, you can copy another file with the same fileref and user number to the disk with PIP only by using the "W" option. However, the BDOS will not reset the Archive attribute bit afterwards, so BU will be unable to recog-

nize the file as changed. The only solution here is to be aware of the problem and, if necessary, to perform a manual file backup immediately after using the utility.

BU accepts ambiguous file references only when a user number is also specified. In one sense, this is an aspect of the user interface design: a user normally should be allowed access to files in one user area only, especially when operations using ambiguous filenames are being performed. More truthfully, BDOS Services 17 (Search for First File) and 18 (Search for Next File) will not accept file references for all user areas. Either they search for all files in all user areas (including erased files), or they search a particular user area only. The only way BU could find an unambiguous or ambiguously speci-

fied file in all user areas would be to search the disk directory 32 times!

### Suggested Enhancements

Those of you with the ambition and the time can extend BU to become a complete file archiving utility. If your system has a hard disk, BU could maintain a catalog file that records each fileref and version number (without erasing previous versions), any pertinent file statistics such as size in kilobytes and assigned file attributes, the backup disk identification number, the time and date the backup copy was made, and the operator who made the backup. Even if you don't own a hard disk, you could maintain a similar sort of file on each backup disk for record-keeping purposes. As a starting point for such a program, you might consid-

er the "Archive" program that is developed in Kernighan & Plauger's book *Software Tools* (Addison-Wesley, ISBN 1-201-03669-X). The program is presented in source code using RATFOR, which for all practical purposes is a variation on a theme of the C programming language.

So, there you have it: a fully functional incremental backup utility for CP/M-80 and CP/M-86. Why DRI did not include an Archive option with PIP for these operating systems after going to the trouble of designing all the necessary features into them is a matter for future historians of computer software to ponder. In the meantime, I hope you enjoy using BU. **DDJ**

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 193.

## Archiving Files Listing

(Text begins on page 36)

```
/* BU.C - A File Backup Utility for CP/M-80 & CP/M-86
*
* Copyright: Ian Ashdown
*             byHeart Software
*             2 - 2016 West First Avenue
*             Vancouver, B.C. V6J 1G8
*             Canada
*
* Acknowledgment: DeSmet C code and suggestions for program
*                   improvement courtesy of Dr. Dobb's Journal
*                   Contributing Editor Anthony SKjellum
*
* Version:      1.1      Written for Aztec CII v1.06b (CP/M-80)
*                 and DeSmet C88 v2.2 (CP/M-86)
*
* Date:         December 31st, 1983 (Version 1.0)
*                 September 7th, 1984 (Version 1.1)
*
* BU utilizes the undocumented "archive" file attribute feature
* of CP/M-80 Versions 2.x and CP/M-86 to automatically detect
* files that have been changed since the disk was last "backed
* up" and copy them (with verification) to a backup disk. This
* program performs the same action as the "A" option of PIP
* under Digital Research's MP/M 2, for which the Archive
* attribute is documented.
*
* Usage: BU x[:afn] y [-AFHQSn]
*
*         where x = drive name of disk to be backed up
*               y = drive name of backup disk
*
*         and the optional arguments are:
*
*               -A      All files, regardless of status
*               -F      Fast copy (without verification)
*               -H      Hard disk (files may be split)
*               -Q      Query each file before backup
*               -S      System attribute copied to backup
```

(Continued on next page)

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
* -n      Backup USER 'n' files only (0-31)
*      afn      Any legal CP/M ambiguous fileref
*      (can only be used with -n option)
*/
/*include "stdio.h"
/*include "ctype.h"      /* Contains macro for "isdigit()" */

/* *** DEFINITIONS ***/

#define AZTEC 1      /* Aztec CII v1.06b (CP/M-80) */
#define DESMET 0      /* DeSmet C88 v2.2 (CP/M-86) */

#if DESMET
#define movmem(src,dest,len) _mov(len,src,dest)
#endif

#define ERROR      -1
#define DEL        -1      /* Deleted fileref flag */
#define ALL        -1      /* All user numbers flag */
#define TRUE       -1
#define FALSE      0
#define SUCCESS    0
#define O_RDONLY   0
#define USER_ERR   0      /* Specified fileref must only be used
                           with -number command-line option */
#define BAD_FREF   1      /* Illegal file reference */
#define BAD_ARGS   2      /* Illegal command line */
#define BAD_OPT    3      /* Illegal option */
#define BAD_USER   4      /* Illegal user number */
#define BAD_DRV    5      /* Illegal drive names */
#define SAME_DRV   6      /* Same drive name for output as input */
#define OPN_ERR    8      /* File open error */
#define READ_ERR   9      /* File read error */
#define CLS_ERR    10     /* File close error */
#define BAD_VFY    11     /* File verify error */
#define DIR_IO     6      /* BDOS Direct I/O service */
#define RESET_DRV  13     /* BDOS Reset All Drives service */
#define SEL_DRV    14     /* BDOS Select Drive service */
#define SRCH_F     17     /* BDOS Search Next service */
#define SRCH_N     18     /* BDOS Search Next service */
#define GET_DRV    25     /* BDOS Get Default Drive service */
#define SET_DMA    26     /* BDOS Set DMA Address service */
#define SET_ATT    30     /* BDOS Set File Attributes service */
#define USER_CODE  32     /* BDOS Get/Set User Code service */
#define MAX_USER   32     /* 32 user codes under CP/M (see DR's
                           documentation for BDOS Service 32) */

/* *** GLOBAL VARIABLES ***/

char ent_drv, /* Entry drive code */
     ent_user, /* Entry user code */
     cur_user; /* Current user code */

/* *** MAIN BODY OF CODE ***/

main(argc,argv)
int argc;
char *argv[];
{
    char in_dsk,          /* Drive name of input disk */
         out_dsk,         /* Drive name of output (backup) disk */
         in_drv,          /* Drive code of input disk */
         out_drv,         /* Drive code of output disk */
         seg_no,          /* Segment number for split files */
         in_file[15],      /* Fileref of current input file */
         out_file[15];     /* Fileref of current output file */
```

(Continued on page 42)

# Dr. Dobbs says, “WE HAVE GOOD NEWS...”

“WINDOWS FOR C can offer you a convenient and reliable means of implementing windows in your text-handling programs...

“The video output is fast and clean, showing no snow with either the graphics or monochrome display.

“WINDOWS FOR C is documented in well-written and readable English, and it appears to be all there.

“WINDOWS FOR C... is a very nice software package that has obviously been well thought out and very cleanly executed.

“WINDOWS FOR C is a professionally-oriented set of programming tools... that we can recommend.”

— Ian Ashdown (*Dr. Dobb's Journal*, November 1984)

## Full support for:

- IBM PC/XT/AT plus compatibles
- All memory models of Lattice C, CI-C86, Mark Wm. C, Aztec C, Microsoft C, Desmet C (PC/MSDOS)

## Plus:

- NEW: Driver interfaces for non-IBM displays
- Full source available

**More than a window display system,** WINDOWS FOR C is a video toolkit for all screen management tasks.

C SOURCE is provided for pop-up menus, viewing of multiple ASCII files within multiple windows, cursor control for vertical and horizontal scrolling, label printing, and text-mode bar graphs.

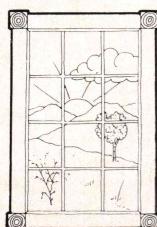
An object-code library of over 50 fully documented building-block subroutines allows you to construct functions to suit your needs.

Once you've used WINDOWS FOR C, you will wonder how you managed without it.

## With WINDOWS FOR C you can:

- Establish any number of windows; clear, write, and control attributes of each window independently; write with word wrap and auto scroll; control all IBM color capabilities.
- Overlay and then restore screens, highlight selected text, format and print with windows, and save windows to memory or disk. Memory management is automatic.

Don't wait. Order now.



The Good News...

# WINDOWS FOR C™

ADVANCED SCREEN MANAGEMENT

WINDOWS FOR C  
(specify compiler & version)

\$195

Demo disk and manual  
(applies toward purchase)

\$ 30

Dealer inquiries welcome

A PROFESSIONAL SOFTWARE TOOL FROM

**CREATIVE SOLUTIONS**

21 Elm Ave., Box D12 • Richford, VT 05476

**802-848-7738**

Master Card & Visa Accepted  
Shipping \$2.50  
VT residents add 4% tax

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
c,          /* Scratch variable */
*s,
*buffer,    /* Scratch string pointer */
/* Pointer to directory entry returned */
/* by "srch_file()" */
*srch_file(),
*malloc();
```

/\* File control blocks of current input and output files \*/

```
static char in_fcb[33],      /* (Automatically initialized */
    out_fcb[33];             /* to zero by compiler) */

/* Structure for linked list of filerefs */

struct file_ref
{
    char name[12];           /* File reference */
    struct file_ref *next;   /* Pointer to next instance */
} root,                      /* Start of linked list */
*fref_1,                     /* Scratch pointers to */
*fref_2;                    /* linked list instances */

/* Initialized file control block for "srch_file()". This FCB
is for a fully ambiguous fileref that causes "srch_file()"
to return all directory entries for the current default
drive. */

static char fcb[] = {'?','?','?','?','?','?','?','?','?',
    '?','?','?','?','?',0,0,0};

int file_cnt = 0,            /* Count of file to be backed up */
    dup_flag,                /* Duplicate fileref flag */
    all_files,                /* All_files flag (cmd-line option) */
    fast_copy,                /* Fast copy flag (cmd-line option) */
    hard_disk,                /* Hard disk flag (cmd-line option) */
    query,                   /* Query flag (cmd-line option) */
    system,                   /* System flag (cmd-line option) */
    user_no,                  /* User number (cmd-line option) */
    next_flag = FALSE; /* Flag to indicate to "srch_file()"
                           that a "search next" is required */

register int i,j;           /* Loop indices */

long begin,                 /* Input file position variables */
end;
```

/\* Display program header \*/

```
printf("\nBU Version 1.1");
printf("                                Copyright 1983, 1984");
printf(" byHeart Software\n\n");

/* Initialize command-line options */

all_files = FALSE;           /* Copy only non-archived files */
fast_copy = FALSE;            /* Copy files with verification */
hard_disk = FALSE;           /* Files will not be split across backup
                             disks if remaining capacity of backup
                             disk is less than current file size */
query = FALSE;                /* Backup without query */
system = FALSE;               /* Assign directory attribute to all
                             backup files */
user_no = ALL;                 /* Backup files in all user areas */

/* Parse command line for user-selected options (if any) */

if(argc < 3)
```

(Continued on page 44)



SOFTWARE™

PRESENTS

CP/M®

FOR THE

Macintosh™

I.Q. SOFTWARE, in one historic move, brings more proven programs to the Macintosh than have existed to date. In the giant world of CP/M based software, **CP/M FOR THE MACINTOSH** delivers the same friendly stand-alone environment to the software developer (professional programmer). If you are wondering why so many developers know and use CP/M it is because CP/M is powerful, easy to learn and easy to use. So easy to use that almost 1 MILLION USERS have CP/M based systems using CP/M based programs and enjoy these same benefits that you will with **CP/M FOR THE MACINTOSH.**



SOFTWARE™

CP/M is a registered trademark of Digital Research, Inc.  
Macintosh is a registered trademark of Apple Computer, Inc.

2229 East Loop 820 North  
Fort Worth, Texas 76118  
(817) 589-2000

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
error(BAD_ARGS,NULL);           /* Illegal command line */
if(argc > 3)
{
    i = 3; /* Start with third command-line argument */
    while(i < argc)
    {
        if(*argv[i] != '-')
            error(BAD_OPT,argv[i]); /* Missing leading '-' */
        s = argv[i]+1;
        while(*s)
        {
            if(*s == 'A')      /* Check for all_files option */
                all_files = TRUE;
            else if(*s == 'F') /* Check for fast copy option */
                fast_copy = TRUE;
            else if(*s == 'H') /* Check for hard disk option */
                hard_disk = TRUE;
            else if(*s == 'Q') /* Check for query option */
                query = TRUE;
            else if(*s == 'S') /* Check for system option */
                system = TRUE;
            else if(isdigit(*s)) /* Check for user number option */
            {
                user_no = *s++ - '0';
                if(isdigit(*s))
                    user_no = user_no * 10 + *s++ - '0';
                if(user_no < 0 || user_no > 31)
                    error(BAD_USER,argv[i]);
                continue;
            }
            else
                error(BAD_OPT,argv[i]);
            s++;
        }
        i++;
    }
}

/* Validate input parameters */

if(*(argv[1]+1) != '\0') /* Check for specified fileref */
{
    if(user_no == ALL)      /* Can only use with specified */
        error(USER_ERR,NULL); /* user number (-n option) */

    /* Modify "fcb[]" to incorporate fileref */

    if(copy_fref(fcb,argv[1]) == ERROR)
        error(BAD_FREF,argv[1]);
}

if(*argv[1] < 'A' || *argv[1] > 'P' ||
   *argv[2] < 'A' || *argv[2] > 'P')
    error(BAD_DRV,NULL); /* Illegal drive names */
if(*argv[1] == *argv[2])
    error(SAME_DRV,NULL); /* Drive names are same */

/* Save entry drive and user codes */

ent_drv = bdos(GET_DRV);
ent_user = bdos(USER_CODE,0xff);

/* Calculate input and output drive codes */

in_drv = (in_dsk = *argv[1]) - 'A' + 1;
out_drv = (out_dsk = *argv[2]) - 'A' + 1;

/* Set default drive to input drive */
```

```

bdos(SEL_DRV,in_drv-1);

/* Set user code to "user_no" if -n option specified */

if(user_no != ALL)
    bdos(USER_CODE,user_no);

/* Read first 12 bytes of updated active directory entries into
   linked list of filerefs. If first byte of entry is 0xe5,
   then file has been erased. */

root.next = NULL;      /* Initialize linked list root */
fref_1 = &root;        /* Initialize linked list pointer */
while(buffer = srch_file(fcb,next_flag))
{
    /* Bit 7 of third filetype byte ('t3') in directory entry
       is the Archive attribute indicator. The BDOS sets this
       bit to zero whenever it updates a directory entry. */

    if(buffer[0] != 0xe5 &&
       (buffer[0] == user_no || user_no == ALL) &&
       (!(buffer[11] & 0x80) || all_files))
    {
        fref_1->next = /* Allocate space for fileref instance */
            (struct file_ref *) malloc(sizeof(struct file_ref));
        fref_1 = fref_1->next; /* Assign space to instance */
        movmem(buffer,fref_1->name,12); /* Move fileref to */
                                         /* linked list instance */
        fref_1->next = NULL; /* Indicate current end of list */
    }
    next_flag = TRUE; /* Only first call to "srch_file()" */
                      /* should be made with "next_flag" */
}
                     /* set to FALSE */

/* If no files have been backed up ... */

if(!root.next) /* Null "root.next" indicates no files have */
{
    /* been changed */
    printf("NO FILES HAVE BEEN UPDATED");
    if(user_no != ALL)
        printf(" in user area %d\n",user_no);
    else
        putchar('\n');
    reset(); /* Reset user and drive codes to entry values */
    exit(0);
}

/* There may be duplicate filerefs in linked list due to some
   files occupying more than one extent on the disk. These
   duplicates must be marked as "deleted" in the list.
   (Duplicate filerefs with different user codes are valid.) */

/* For all filerefs ... */

fref_1 = &root; /* Initialize a linked list pointer */
while(fref_1->next)
{
    fref_1 = fref_1->next; /* Root instance is NULL entry */
    dup_flag = FALSE; /* Reset duplicate fileref flag */

    /* For all preceding filerefs ... */

    fref_2 = &root; /* Initialize another linked list pointer */
    fref_2 = fref_2->next; /* Skip root instance */
    while(fref_2->next != fref_1->next)
    {
        /* Compare filerefs (ignore deleted filerefs and different
           user codes) */
    }
}

```

(Continued on next page)

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
if(fref_2->name[0] != DEL && fref_1->name[0] == fref_2->name[0])
    if(!strcmp(fref_1->name+1,fref_2->name+1,11))
    {
        dup_flag = TRUE;      /* Indicate duplicate fileref */
        break;
    }
    fref_2 = fref_2->next;
}
if(dup_flag == TRUE)
    fref_1->name[0] = DEL;    /* Delete if duplicate fileref */
else
    file_cnt++;            /* Increment file count */
}

/* Display file copy header */

printf("Number of files to be copied: %d\n\n",file_cnt);
printf("User:    Files being copied to Drive %c:\n\n",
       out_dsk);

/* Initialize current input and output fileref templates */

in_file[0] = in_dsk;
out_file[0] = out_dsk;
in_file[1] = out_file[1] = ':';
in_file[10] = out_file[10] = '.';
in_file[14] = out_file[14] = '\0';

/* Initialize current input and output FCB templates */

in_fcb[0] = in_drv;
out_fcb[0] = out_drv;

/* For all validated filerefs do ... */

for(cur_user = 0; cur_user < MAX_USER; cur_user++)
{
    if(user_no != ALL)
        if(cur_user != user_no)
            continue;
    bdos(USER_CODE,cur_user); /* Set user code to "cur_user" */
    fref_1 = &root;          /* Initialize linked list pointer */
    while(fref_1->next)
    {
        fref_1 = fref_1->next; /* Root instance is NULL entry */
        if(fref_1->name[0] == cur_user)
        {
            /* Update the current input and output FCB's */

            movmem(fref_1->name+1,in_fcb+1,11);
            movmem(fref_1->name+1,out_fcb+1,11);

/* Reset the Read-Only and System attribute bits of the
   FCB's so that the file can be copied and displayed
   (unless the "system" flag is TRUE) */

out_fcb[9] &= 0x7f;      /* Read-Only attribute */
if(!system)
    out_fcb[10] &= 0x7f; /* System attribute */

/* Set the Archive attribute bit of the FCB's to indicate
   that the file has been backed up */

in_fcb[11] |= 0x80;
out_fcb[11] |= 0x80;
```

(Continued on page 48)

# DIGITAL RESEARCH COMPUTERS

(214) 225-2309



## IBM PC ADD ON BOARDS NOW IN STOCK!

**RIO PLUS II™** — Multi-function board with 2 RS232 serial ports (1 standard, 1 optional), parallel I/O port, game port and clock/calendar. Comes with 64K memory (expandable to 384K) and PC Accelerator.

STB-RIO + II 64 List \$395 \$286.00

### GRAPHIX PLUS II™

Multi-function video board featuring monochrome text, full screen monochrome graphics, RGB color, composite b&w video, a parallel port, a light pen interface, and an upgradable clock option.

STB-GX List \$495 \$363.00

### SUPER RIO™

RAM-I/O multi-function board with 64K memory (upgradable to 768K), two serial ports, parallel printer port, game port and clock/calendar.

STB-SRIO-64 List \$419 \$294.00

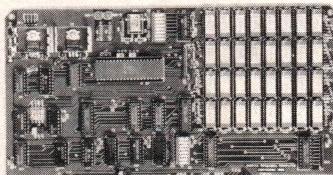
All above boards come with "PC Accelerator" which is a RAM Disk Emulator and Printer Spooler that vastly increases your processing speed. All boards fully assembled and tested, and are warranted by the manufacturer for 1 year!

**SPECIAL OFFER:** Buy any two or more of the above boards and take an additional \$10 per board discount

### 256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

### PRICE CUT!



BLANK PCB  
(WITH CP/M® 2.2  
PATCHES AND INSTALL  
PROGRAM ON DISKETTE)  
**\$69.95**  
(8203-1 INTEL \$29.95)

**\$259.00**

#LS-100 (FULL 256K KIT)

#### FEATURES:

- \* 256K on board, using +5V 64K DRAMS.
- \* Uses new Intel 8203-1 LSI Memory Controller.
- \* Requires only 4 Dip Switch Selectable I/O Ports.
- \* Runs on 8080 or Z80 S100 machines.
- \* Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- \* Provisions for Battery back-up.
- \* Software to mate the LS-100 to your CP/M® 2.2 DOS is supplied.
- \* The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- \* Compare our price! You could pay up to 3 times as much for similar boards.

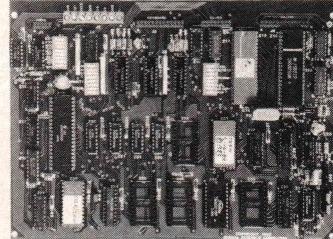
### THE NEW ZRT-80

### CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

#### FEATURES:

- \* Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- \* RS232 at 16 BAUD Rates from 75 to 19,200.
- \* 24 x 80 standard format (60 Hz).
- \* Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- \* Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- \* Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- \* 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- \* Composite or Split Video.
- \* Any polarity of video or sync.
- \* Inverse Video Capability.
- \* Small Size: 6.5 x 9 inches.
- \* Upper & lower case with descenders.
- \* 7 x 9 Character Matrix.
- \* Requires Par. ASCII keyboard.



BLANK PCB WITH 2716

CHAR. ROM, 2732 MON. ROM

**\$49.95**

SOURCE DISKETTE - ADD \$10  
SET OF 2 CRYSTALS - ADD \$7.50

WITH 8 IN.  
SOURCE DISK!  
(CP/M COMPATIBLE)

**\$99.95**

(COMPLETE KIT,  
2K VIDEO RAM)

## Digital Research Computers

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

### 64K S100 STATIC RAM

**\$179.00**  
KIT

### NEW!

LOW POWER!

150 NS ADD \$10

BLANK PC BOARD  
WITH DOCUMENTATION  
\$49.95

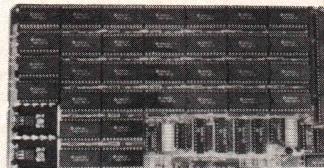
SUPPORT IC'S + CAPS  
\$17.50

FULL SOCKET SET  
\$14.50

FULLY SUPPORTS THE  
NEW IEEE 696 S100  
STANDARD  
(AS PROPOSED)

FOR 56K KIT \$165

ASSEMBLED AND  
TESTED ADD \$50



#### FEATURES: PRICE CUT!

- \* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- \* Fully supports IEEE 696 24 BIT Extended Addressing.
- \* 64K draws only approximately 500 MA.
- \* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- \* SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- \* 2716 EPROMs may be installed in any of top 48K.
- \* Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- \* Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- \* BOARD may be partially populated as 56K.

### 64K SS-50 STATIC RAM

**\$149.95**  
(48K KIT)

### NEW!

LOW POWER!

RAM OR EPROM!

BLANK PC BOARD  
WITH  
DOCUMENTATION  
\$52

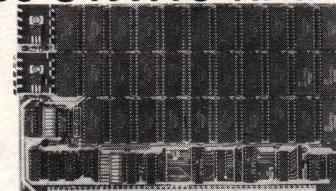
SUPPORT IC'S + CAPS  
\$18.00

FULL SOCKET SET  
\$15.00

56K Kit \$169

64K Kit \$199

ASSEMBLED AND  
TESTED ADD \$50



- \* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- \* Fully supports Extended Addressing.
- \* 64K draws only approximately 500 MA.
- \* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- \* Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- \* 2716 EPROMs may be installed anywhere on Board.
- \* Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- \* One Board supports both RAM and EPROM.
- \* RAM supports 2MHZ operation at no extra charge!
- \* Board may be partially populated in 16K increments.

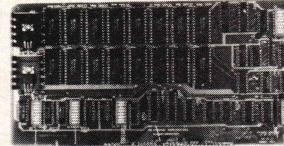
### 32K S100 EPROM/STATIC RAM

### NEW!

### FOUR FUNCTION BOARD!

### NEW!

EPROM II  
FULL  
EPROM KIT  
\$69.95  
A&T EPROM  
ADD \$35.00



BLANK  
PC BOARD  
WITH DATA  
\$39.95

SUPPORT  
IC'S  
PLUS CAPS  
\$16

FULL  
SOCKET SET  
\$15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

PRICES  
SLASHED!

- \* This one board can be used in any one of four ways:
  - A. As a 32K 2716 EPROM Board
  - B. As a 32K 2732 EPROM Board (Using Every Other Socket)
  - C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
  - D. As a 32K Static RAM Board
- \* Uses New 2K x 8 (TMM 2016 or HM 6116) RAM's
- \* Fully Supports IEEE 696 Bus Standard (As Proposed)
- \* Supports 24 Bit Extended Addressing
- \* 200 NS (FAST!) RAM's are standard on the RAM Kit
- \* Supports both Cromemco and North Star Bank Select
- \* Supports Phantom
- \* On Board wait State Generator
- \* Every 2K Block may be disabled
- \* Addressed as two separate 16K Blocks on any 64K Boundary
- \* Perfect for MP/M® Systems
- \* RAM Kit is very low power (300 MA typical)

### 32K STATIC RAM KIT — \$109.95

For RAM Kit A&T — Add \$400

**TERMS:** Add \$3.00 postage. We pay balance. Orders under \$15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCharge. Tex. Res. add 1 1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50, add 85¢ for insurance.

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
/* Move the fileref from the FCB's to the initialized
   input and output fileref templates */

movmem(in_fcb+1,in_file+2,8); /* Filename move */
movmem(out_fcb+1,out_file+2,8);
movmem(in_fcb+9,in_file+11,3); /* Filetype move */
movmem(out_fcb+9,out_file+11,3);

/* Strip high order bits off filerefs to form proper
   ASCII characters */

for(j = 2; j <= 13; j++)
{
    in_file[j] &= 0x7f;
    out_file[j] &= 0x7f;
}

/* Display the filerefs */

printf(" %2d      %s --> %s",cur_user,in_file,out_file);

/* Query operator for backup if indicated by "query"
   flag */

if(query)
{
    printf(" O.K. to backup?  ");
    if((c = in_chr()) == 'y' || c == 'Y')
        puts("Yes");
    else
    {
        puts("No");
        continue; /* Go do next fileref if "No" */
    }
}
else
    putchar('\n');

/* Copy file from the input disk to the output disk */

if(hard_disk) /* Split file across backup disks if */
{             /* necessary */
    begin = OL; /* Initialize file position pointer */
    seg_no = 0; /* and split file segment number */
    do
    {
        /* Reset the Read-Only attribute of the output file
           (if it exists) so that the input file can be
           copied to it */

        bdos(SET_ATT,out_fcb);

        end = copy_file(in_file,out_file,begin);
        if(!fast_copy) /* Verify file unless -F selected */
            verify_file(in_file,out_file,begin);

        /* Set the Read-Only attribute of the output file */

        out_fcb[9] != 0x80;
        bdos(SET_ATT,out_fcb);
        out_fcb[9] &= 0x7f; /* ... and reset the fcb */
        if(end != NULL)
        {
            /* File has been partially written on current
               backup disk - new disk required to continue */
        }
    }
}
```

```

new_disk(out_file,hard_disk);

/* Append segment number to filename of output
   fileref (e.g. - B:FILE.TYP will become
   B:FILE--01.TYP) */

seg_no++;
for(j = 2; j <= 7; j++) /* Change spaces to */
    if(out_file[j] == ' ') /* '-' character */
        out_file[j] = '-';
out_file[8] = seg_no/10 + '0'; /* Append segment */
out_file[9] = seg_no%10 + '0'; /* number */

/* Display filerefs again */

printf(" %2d      %s --> %s\n",
       cur_user,in_file,out_file);
}

begin = end;
}
while(end != NULL); /* Loop until file is written */
}
else
{
/* Reset the Read-Only attribute of the output file
   (if it exists) */

bdos(SET_ATT,out_fcb);

if(copy_file(in_file,out_file,OL) != NULL)
{
/* Disk was full - erase partially written file, back
   up fileref pointer and rewrite file to new disk */

unlink(out_file);
new_disk(out_file,hard_disk);
i--;
continue;
}

if(!fast_copy) /* Verify file unless -F selected */
    verify_file(in_file,out_file,OL);

/* Set the Read-Only attribute of the output file */

out_fcb[9] |= 0x80;
bdos(SET_ATT,out_fcb);
}

/* Set the Archive attribute of the input file to
   indicate that the file was successfully backed up */

bdos(SET_ATT,in_fcb);
}
}
}

reset(); /* Reset user and drive codes to entry values */
}

/*** FUNCTIONS **/


/* Search for first or next directory entry */

char *srch_file(fcb_ptr,next_flag)
char *fcb_ptr;           /* Pointer to file control block */
int next_flag;           /* Flag to indicate "search next" */
{
    static char sf_cur[32], /* Current directory entry buffer */
               sf_fcb[36]; /* File control block buffer */

```

(Continued on next page)

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
int index, /* Index of directory entry in DMA buffer */
    *ptr; /* Pointer to directory entry in DMA buffer */

bdos(SET_DMA,0x80); /* Set DMA address to 80h */
if(!next_flag)
{
    movmem(fcb_ptr,sf_fcb,16); /* Initialize FCB buffer */
    if((index = bdos(SRCH_F,sf_fcb)) == 0xff) /* Find first */
        return NULL; /* Return NULL if unsuccessful */
}
else
    if((index = bdos(SRCH_N,NULL)) == 0xff) /* Find next */
        return NULL; /* Return NULL if unsuccessful */

/* BDOS services 17 and 18 leave four consecutive directory
   entries of 32 bytes each in the 128-byte DMA buffer and also
   returns an index value of 0, 1, 2 or 3 to indicate the
   correct directory entry in the accumulator. The "bdos()"
   function returns this index value. */

ptr = 0x80 + index * 32; /* Calculate pointer to directory
                           entry */
movmem(ptr,sf_cur,32); /* Move directory entry to current
                           directory entry buffer */
return sf_cur;
}

/* Copy file starting at "offset" from beginning */

copy_file(in_file,out_file,offset)
char *in_file,           /* Input fileref */
     *out_file;          /* Output fileref */
long offset;            /* Input file position offset */
{
    register int in_cnt, /* Character counts for unbuffered I/O */
                out_cnt;
    int fd_in,           /* Input file descriptor */
        fd_out,           /* Output file descriptor */
        full_disk = FALSE; /* Full disk flag */

    char *buffer,         /* Input file buffer */
        *buff_ptr, /* Pointer to current position in "buffer[]" */
        *malloc();

    unsigned buf_size = 32768; /* Initial memory allocation size */

/* "read()" accepts a maximum of 32768 bytes at a time. Allocate
   as much memory as possible up to this limit for the input
   buffer, using 128 byte decrements. */

do
    if(buffer = malloc(buf_size))
        break;
    while(buf_size -= 128);

/* Open input file for unbuffered Read-Only access */

if((fd_in = open(in_file,O_RDONLY)) == ERROR)
    error(OPN_ERR,in_file);

/* Create the output file by first deleting it (if it
   exists), then opening it for unbuffered Write-Only
   access. */
```

(Continued on page 52)

# B. G. MICRO

P. O. Box 280298 Dallas, Texas 75228  
 (214) 271-5546



## 74LS

| LS00  | .25  | LS162    | .65  |
|-------|------|----------|------|
| LS01  | .24  | LS163    | .50  |
| LS02  | .25  | LS164    | .65  |
| LS03  | .25  | LS165    | .90  |
| LS04  | .25  | LS166    | .99  |
| LS05  | .25  | LS169    | 1.25 |
| LS08  | .25  | LS174    | .50  |
| LS09  | .25  | LS175    | .50  |
| LS10  | .24  | LS181    | 1.99 |
| LS11  | .24  | LS191    | .90  |
| LS12  | .30  | LS192    | .80  |
| LS13  | .40  | LS193    | .80  |
| LS14  | .50  | LS194    | .65  |
| LS15  | .32  | LS195    | .65  |
| LS20  | .25  | LS196    | .75  |
| LS21  | .28  | LS197    | .85  |
| LS27  | .28  | LS221    | .75  |
| LS30  | .25  | LS240    | 1.00 |
| LS32  | .28  | LS241    | .80  |
| LS33  | .50  | LS242    | 1.00 |
| LS37  | .33  | LS243    | 1.00 |
| LS38  | .34  | LS244    | 1.00 |
| LS42  | .45  | LS245    | 1.20 |
| LS51  | .24  | LS251    | .50  |
| LS54  | .25  | LS253    | .55  |
| LS55  | .27  | LS257    | .55  |
| LS73  | .35  | LS258    | .55  |
| LS74  | .35  | LS259    | 2.50 |
| LS85  | .65  | LS260    | .50  |
| LS86  | .30  | LS266    | .50  |
| LS90  | .50  | LS273    | 1.25 |
| LS93  | .55  | LS279    | .45  |
| LS97  | 2.00 | LS280    | 1.50 |
| LS107 | .37  | LS283    | .60  |
| LS109 | .35  | LS290    | .85  |
| LS112 | .33  | LS293    | .85  |
| LS122 | .45  | LS298    | .89  |
| LS123 | .65  | LS299    | 1.60 |
| LS124 | 2.75 | LS323    | 2.60 |
| LS125 | .50  | LS348    | .75  |
| LS126 | .49  | LS366    | .45  |
| LS132 | .50  | LS367    | .50  |
| LS133 | .45  | LS368    | .40  |
| LS138 | .50  | LS373    | 1.25 |
| LS139 | .50  | LS374    | 1.30 |
| LS151 | .50  | LS375    | 1.00 |
| LS153 | .50  | LS377    | 1.49 |
| LS154 | 1.50 | LS378    | .85  |
| LS155 | .50  | LS390    | 1.19 |
| LS156 | .55  | LS393    | 1.19 |
| LS157 | .50  | LS399    | 1.25 |
| LS158 | .50  | LS670    | 1.50 |
| LS160 | .65  | 25LS2569 | 3.00 |
| LS161 | .55  |          |      |

## TTL

| 7400  | .22 | 74121 | .27  |
|-------|-----|-------|------|
| 7402  | .22 | 74123 | .40  |
| 7408  | .24 | 74125 | .49  |
| 7410  | .19 | 74151 | .50  |
| 7413  | .33 | 74153 | .50  |
| 7420  | .22 | 74154 | 1.19 |
| 7425  | .25 | 74157 | .50  |
| 7427  | .25 | 74160 | .80  |
| 7428  | .25 | 74162 | .60  |
| 7432  | .27 | 74163 | .60  |
| 7433  | .25 | 74164 | .80  |
| 7440  | .19 | 74165 | .80  |
| 7442  | .40 | 74166 | 1.00 |
| 7451  | .20 | 74173 | .75  |
| 7473  | .34 | 74174 | .85  |
| 7474  | .40 | 74175 | .80  |
| 7483  | .45 | 74185 | 1.70 |
| 7485  | .55 | 74192 | .70  |
| 7486  | .30 | 74193 | .80  |
| 7490  | .35 | 74195 | .80  |
| 7493  | .33 | 74273 | 1.75 |
| 7496  | .40 | 74367 | .60  |
| 74107 | .28 | 74390 | 1.40 |
| 74109 | .45 |       |      |

## EPROM

|                        |       |
|------------------------|-------|
| 2708 1KX8 450 n.s.     | 2.20  |
| 2758 1KX8 +5V 450 n.s. | 2.50  |
| 2716 2KX8 450 n.s.     | 3.20  |
| 2716-1 2KX8 350 n.s.   | 4.95  |
| 2732 4KX8 450 n.s.     | 4.00  |
| 2732A-3                | 5.70  |
| 2732A-35               | 5.20  |
| 2532 4KX8 450 n.s.     | 3.00  |
| 2764-25                | 6.00  |
| 2764-3                 | 5.00  |
| 27128-25               | 15.50 |
| 27128-3                | 13.50 |
| 27128-45               | 12.00 |

## EPROM SPECIAL

We bought a large quantity of 2708s from a computer manufacturer who redesigned their boards. We removed them from sockets, erased and verified them, and now we offer the savings to you. Complete satisfaction guaranteed.

\$1.49 or 10/\$12.00

## STATIC RAM

|                         |         |
|-------------------------|---------|
| 2016-2KX8 200 n.s.      | 8/29.95 |
| 2101-1 - 256X4 500 n.s. | .75     |
| 21L02-1 350 n.s.        | .65     |
| 2102AL-4 L.P. 450 n.s.  | .49     |
| 2111-1 256X4 500 n.s.   | 2.00    |
| 2114L-3 1KX4 300 n.s.   | 1.50    |
|                         | 8/10.00 |
| 2125A-2 1KX1 70 n.s.    | 2.20    |
| 2142-3 1KX4 300 n.s.    | 1.50    |
| TMS4044 (MCM6641 C-25)  |         |
| 4KX1 250 n.s.           | 8/6.00  |
| 5101 - 256X4 - CMOS     | \$1.00  |
| 6116P-4-2KX8            |         |
| 200 n.s. CMOS           | 8/29.95 |
| 6501-5 256X4 - CMOS     | 1.00    |

## 4K STATIC RAMS

LESS THAN 50¢ EACH  
 MK4104J-4 - 250 N.S. 18 Pin Ceramic Computer Mfg.  
 Surplus. PRIME. Fully Static. Easy to Use. Has Same Pin Out as TMS4044, but slightly different timing. With Specs. (Mostek)

8 for 5.00 32 for 15.95  
 VERY LOW POWER!

## DYNAMIC RAM

|                             |          |
|-----------------------------|----------|
| 2108-4 8KX1                 | 1.50     |
| 2118-4 16KX1-5Volt          | 1.50     |
| 4027-4KX1-250 n.s.          | .80      |
| 4116-16KX1-250 n.s.         | 8/8.00   |
| 4116-16KX1-200 n.s.         | 8/11.50  |
| 4116 16KX1-150 n.s.         | 8/12.95  |
| 4164 -+5V 64K 200 n.s.      | 8/32.00  |
| 4164 150 n.s.               | 8/34.00  |
| TMS4416-16KX4-150 n.s.      | 5.25     |
| MK4516-15 16KX1-5Volt       | 1.50     |
| 5280N-5 (2107B-4 • TMS4060) |          |
| 4KX1                        | 8/3.95   |
| 41256 150 n.s.              | 8/199.00 |

## SPECIAL

|                     |        |
|---------------------|--------|
| AY3-8910            |        |
| W/60 Page Manual    |        |
| New Price — \$7.00  |        |
| 5832 Clock-Calendar | \$3.95 |

## 8000

|       |       |        |        |
|-------|-------|--------|--------|
| Z8002 | 20.00 | 8085A  | 5.95   |
| 8035  | 4.50  | 8086-2 | 24.95  |
| 8039  | 4.50  | 8087-3 | 159.00 |
| 8080A | 1.25  | 8088   | 15.00  |

## 8200

|         |       |        |       |
|---------|-------|--------|-------|
| 8202A   | 15.95 | 8251   | 4.20  |
| D8203-1 | 29.95 | 8255-5 | 5.00  |
| 8212    | 1.50  | 8257   | 6.00  |
| 8214    | 2.00  | 8259A  | 3.50  |
| 8216    | 1.75  | 8275   | 19.95 |
| 8228    | 3.25  | 8284   | 3.20  |
| 8237-5  | 7.50  | 8287   | 5.75  |
| 8250B   | 9.95  | 8288   | 7.50  |

## SOCKETS

|                         |         |
|-------------------------|---------|
| Low Profile SOLDER TAIL |         |
| 6 Pin                   | 14/1.00 |
| 8 Pin                   | 13/1.00 |
| 14 Pin                  | 10/1.00 |
| 16 Pin                  | 8/1.00  |
| 18 Pin                  | 8/1.00  |
| 20 Pin                  | 7/1.00  |
| 22 Pin                  | 7/1.00  |
| 24 Pin                  | 6/1.00  |
| 28 Pin                  | 6/1.00  |
| 40 Pin                  | 5/1.00  |

BUY \$10  
 GET \$1.00 - FREE CHOICE

## 2114 SPECIAL!

|   |      |
|---|------|
| COMPUTER MANUFACTURERS EXCESS INVENTORY SALE! |      |
| PRIME! 2114-300 n.s.                          |      |
| INCREDIBLE PRICE!                             |      |
| YOU SAVE!                                     |      |
| 8/\$8.00                                      |      |
| GUARANTEED                                    |      |
| CRYSTALS                                      |      |
| 32.768 KHz SPECIAL                            | .65  |
| 262.144                                       | .75  |
| 300.000                                       | 1.00 |
| 307.200                                       | 1.25 |
| 1.5432 Mhz                                    | .75  |
| 1.8432  | 2.49 |
| 2.000000                                      | 2.49 |
| 2.560   | 1.49 |
| 3.000   | 1.15 |
| 3.120   | 1.20 |
| 3.2   | 1.49 |
| 3.4560  | 1.49 |
| 3.579545                                      | 1.00 |
| 4.000   | 2.49 |
| 4.194304                                      | 1.50 |
| 4.433618                                      | .75  |
| 4444.000                                      | 1.25 |
| 4.9152  | 2.49 |
| 4.916 Bd. Rate                                | 1.25 |
| 5.000000                                      | 1.50 |
| 5.0688  | 3.75 |
| 5.616   | 1.59 |
| 6.000   | 2.49 |
| 6.176   | 1.49 |
| 7.164112                                      | 1.49 |
| 7.3728  | 1.49 |
| 8.000   | 1.99 |
| 9.000   | 1.49 |
| 9.90000                                       | 1.25 |
| 10.69425                                      | 3.75 |
| 10.8864                                       | 1.49 |
| 10.920  | 1.49 |
| 11.088  | 1.59 |
| 12.000  | 2.75 |
| 13.440  | 1.00 |
| 14.31818                                      | 2.49 |
| 15.2  | 1.10 |
| 16.00000                                      | 1.50 |
| 16.5888                                       | 1.49 |
| 17.430  | 2.49 |
| 18.2259                                       | 1.00 |
| 20.000  | 3.75 |
| 21.87108                                      | 1.00 |
| 22.092  | 1.00 |
| 32.000  | 2.49 |
| 40.000  | 2.00 |
| 87.3333                                       | 1.00 |
| 91.000  | 1.00 |
| 104.8   | 1.00 |

## CONTROLLER SET

## UART

|                             |      |
|-----------------------------|------|
| TR1602B (COM 2017)          | 1.75 |
| IM6402-(1863)+5v High speed |      |
| AY5-1013 pin out            | 2.95 |
| INS 8250B                   | 9.95 |

## 6500

|      |      |      |      |
|------|------|------|------|
| 6502 | 2.60 | 6545 | 7.50 |
| 6522 | 6.95 | 6551 | 6.50 |

## SPECIALS

|         |      |
|---------|------|
| 1408L8  | 2.00 |
| BR1941L | 5.95 |
| LM311   | .40  |
| LF353   | .60  |
| LF356   | .60  |
| TL494   | 2.50 |
| LM555   | .30  |
| 733     | .60  |
| 9401    | 5.00 |
| DS8835  | 1.50 |
| MC4024P | 2.50 |
| NE592   | 1.50 |
| LM319   | 1.00 |
| MC1350  | 1.00 |
| LM2917  | 1.50 |
| LM339   | .50  |

TERMS: (Unless specified elsewhere) Add \$1.50 postage, we pay balance. Orders over \$50.00 add 85¢ for insurance. No C.O.D. Texas Res. add 6-1/8% Tax. 90 Day Money Back Guarantee on all items. All items subject to prior sale. Prices subject to change without notice. Foreign order - US funds only. We cannot ship to Mexico. Countries other than Canada, add \$3.50 shipping and handling.

Circle no. 13 on reader service card.

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
if((fd_out = creat(out_file,NULL)) == ERROR)
    error(OPN_ERR,out_file);

/* Initialize input file position pointer to "offset" */

lseek(fd_in,offset,0);

/* Copy input file to output file by buffering data
   through "buffer[]" */

do
{
    if((in_cnt = read(fd_in,buffer,buf_size)) == ERROR)
        error(READ_ERR,in_file);
    buff_ptr = buffer; /* Initialize "buffer[]" pointer */
    out_cnt = 0;         /* and "out_cnt" */
    do
    {
        /* Write contents of "buffer[]" to output file in 128
           byte records until either the buffer is written or a
           write error occurs. Since the Ox1a (^Z) character CP/M
           uses as an EOF marker is a valid file character for non-
           ASCII files, "read()" always reads the last 128 byte
           record of a file under CP/M. */

        if(write(fd_out,buff_ptr,128) != 128)
        {
            /* The standard implementation of "write()" does not
               distinguish between a full disk or directory and a
               write error in its returned error code. Thus, it is
               assumed that an error means a full disk/directory. */

            full_disk = TRUE;
            break;
        }
        buff_ptr += 128; /* Increment "buffer[]" ptr */
        out_cnt += 128; /* Update count of chars written */
    }
    while(in_cnt > out_cnt); /* Until end of "buffer[]" */
    offset += out_cnt; /* Update input file position pointer */
    if(full_disk == TRUE)
        break;
}
while(in_cnt == buf_size); /* Until end of file */
free(buffer); /* Deallocate buffer space */
if(close(fd_in) == ERROR) /* Close the files */
    error(CLS_ERR,in_file);
if(close(fd_out) == ERROR)
    error(CLS_ERR,out_file);

/* If full disk return new offset for input file, else */
/* return NULL to indicate completion of file copy operation */

return (full_disk ? offset : NULL);
}

/* Compare portion of input file starting at "offset" from begin-
   ning of file with output file */

verify_file(in_file,out_file,offset)
char *in_file,             /* Input fileref */
     *out_file;            /* Output fileref */
long offset;               /* Input file position offset */
{
    register int match_cnt; /* Scratch variable */
```

```

int out_cnt, /* Character counts for unbuffered I/O */
    fd_in, /* Input file descriptor */
    fd_out; /* Output file descriptor */

char *buffer, /* Dynamically-allocated buffer */
    *in_ptr, /* Input file buffer pointer */
    *out_ptr, /* Output file buffer pointer */
    *malloc();

unsigned buf_size = 65280; /* Initial memory allocation size */

/* "read()" and "write()" accept a maximum of 32768 bytes at a
   time. Allocate as much memory as possible up to this limit
   for both the input and output buffers, using 256 byte
   decrements (128 bytes for each buffer). */

do
    if(buffer = malloc(buf_size))
        break;
while(buf_size -= 256);

/* Divide "buffer[]" in two for "in_ptr" and "out_ptr" */

buf_size /= 2;

if((fd_in = open(in_file,O_RDONLY)) == ERROR) /* Open files */
    error(OPN_ERR,in_file);
if((fd_out = open(out_file,O_RDONLY)) == ERROR)
    error(OPN_ERR,out_file);
lseek(fd_in,offset,0); /* Initialize file position pointer */

/* Read in characters from both files and compare */

do
{
    in_ptr = buffer; /* Assign buffer pointers */
    out_ptr = in_ptr + buf_size;
    if(read(fd_in,in_ptr,buf_size) == ERROR)
        error(READ_ERR,in_file);
    if((out_cnt = read(fd_out,out_ptr,buf_size)) == ERROR)
        error(READ_ERR,out_file);
    match_cnt = out_cnt;
    while(match_cnt--)
        /* Verify character */
        /* by character, and */
        /* delete the output */
        if(close(fd_out) == ERROR)
            error(CLS_ERR,out_file);
        unlink(out_file);
        error(BAD_VFY,out_file);
    }
    while(out_cnt == buf_size); /* Until end of output file */
    free(buffer); /* Deallocate buffer space */
    if(close(fd_in) == ERROR) /* Close the files - verifica- */
        error(CLS_ERR,in_file); /* tion was successful */
    if(close(fd_out) == ERROR)
        error(CLS_ERR,out_file);
}

/* Copy fileref to file control block */

copy_fref(fcb,fref)
char *fcb, /* Pointer to file control block */
      *fref; /* Pointer to fileref */
{
    char c; /* Scratch variable */
    int i, /* Fileref index variable */
        j, /* FCB index variable */
        k, /* Scratch variable */
        done; /* Loop break flag */

```

(Continued on next page)

# Archiving Files Listing (Listing Continued, text begins on page 36)

```
if(fref[1] != ':' || fref[2] == '\0')
    return ERROR; /* No drive code separator or null fileref */

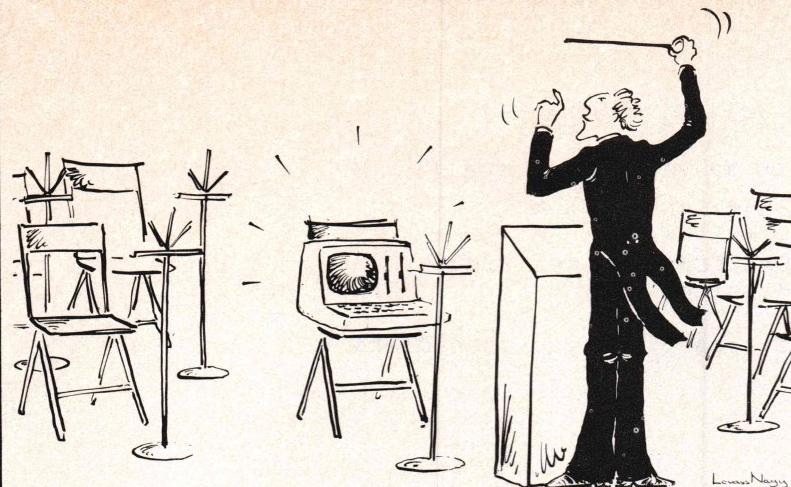
/* Calculate drive code from drive name and put in FCB */

fcb[0] = fref[0] - 'A' + 1;

/* Process remainder of fileref */

done = FALSE;
for(i = 2,j = 1;i <= 9;i++,j++)           /* Skip drive code in */
{                                         /* fileref */
    switch(c = fref[i])
    {
        case '.':             /* Filetype separator */
            if(i == 2)
                return ERROR; /* Null filename */
            for( ; j <= 8; j++)
                fcb[j] = ' '; /* Pad filename with trailing blanks */
            done = TRUE;
            break;
        case '*':             /* Match any following string */
            for( ; j <= 8; j++)
                fcb[j] = '?'; /* Pad filename with trailing */
            i++;
            /* question marks */
            done = TRUE;
            break;
        case '\0':             /* End of fileref */
            for( ; j <= 11; j++) /* Pad FCB with trailing spaces */
                fcb[j] = ' ';
            return SUCCESS;
        case ',':              /* Illegal filename characters */
        case ';':
        case ':':
        case '=':
        case '[':
        case ']':
        case '_':
        case '<':
        case '>':
            return ERROR;
        default:
            if(c >= '!') && c <= '~')
                fcb[j] = c; /* Copy character from fileref to FCB */
            else
                return ERROR; /* Nonprintable character or ' ' */
    }
    if(done)
        break;
}
c = fref[i];
if((c = fref[i]) == '\0')      /* End of fileref */
{
    for( ; j <= 11; j++) /* Pad FCB with trailing spaces */
        fcb[j] = ' ';
    return SUCCESS;
}
else if(c == '.')           /* Filetype separator */
{
    i++;
    k = i + 2;             /* Set limit of 3 characters */
    for( ; i <= k; i++,j++)
    {
        done = FALSE;
        switch(c = fref[i])
```

(Continued on page 56)



Would you hire an entire band when all you need is one instrument? Of course not.

So why use a whole orchestra of computers when all you need is one to develop software for virtually any type of micro-processor?

The secret? Avocet's family of cross-assemblers. With Avocet cross-assemblers you can develop software for practically every kind of processor — without having to switch to another development system along the way!

## Cross-Assemblers to Beat the Band!

### Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 4 years of actual use. Ask NASA, IBM, Xerox or the hundreds of other organizations that use them. Every time you see a new microprocessor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on almost any personal computer and process assembly language for the most popular microprocessor families.

### Your Computer Can Be A Complete Development System

Avocet has the tools you need to enter and assemble your software and finally cast it in EPROM:

**VEDIT Text Editor** makes source code entry a snap. Full-screen editing plus a TECO-like command mode for advanced tasks. Easy installation - INSTALL program supports over 40 terminals and personal computers. Customizable keyboard layout. CP/M-80, CP/M-86, MSDOS, PCDOS.....\$150

**EPROM Programmers** let you program, verify, compare, read, display EPROMS but cost less because they communicate through your personal computer or terminal. No personality modules! On-board intelligence provides menu-based setup for 34 different EPROMS, EEPROMS and MPUs (40-pin devices require socket adaptors). Self-contained unit with internal power supply, RS-232 interface, Textool ZIF socket. Driver software (sold separately) gives you access to all programmer features through your computer, lets you download cross-assembler output files, copy EPROM to disk.

### Model 7228 Advanced Programmer

—Supports all PROM types listed. Super-fast "adaptive" programming algorithm programs 2764 in 1.1 minutes.

### Model 7128 Standard Programmer

—Lower-cost version of 7228. Supports all PROM types except "A" versions of 2764 and 27128. Standard programming algorithm programs 2764 in 6.8 minutes.

| Avocet Cross-assembler | Target Microprocessor | CP/M-80   | CP/M-86<br>IBM PC, MSDOS** |
|------------------------|-----------------------|-----------|----------------------------|
| XASM04 NEW             | 6804                  | \$ 250.00 | \$ 250.00                  |
| XASM05                 | 6805                  | 200.00    | 250.00                     |
| XASM09                 | 6809                  | 200.00    | 250.00                     |
| XASM18                 | 1802/1805             | 200.00    | 250.00                     |
| XASM48                 | 8048/8041             | 200.00    | 250.00                     |
| XASM51                 | 8051                  | 200.00    | 250.00                     |
| XASM65                 | 6502/65C02            | 200.00    | 250.00                     |
| XASM68                 | 6800/01, 6301         | 200.00    | 250.00                     |
| XASM75                 | NEC 7500              | 500.00    | 500.00                     |
| XASM85                 | 8085                  | 250.00    | 250.00                     |
| XASM400                | COP400                | 300.00    | 300.00                     |
| XASMF8                 | F8/3870               | 300.00    | 300.00                     |
| XASMZ8                 | Z8                    | 200.00    | 250.00                     |
| XASMZ80                | Z80                   | 250.00    | 250.00                     |
| XMAC682 NEW            | 68200                 | 595.00    | 595.00                     |
| XMAC68K NEW            | 68000/68010           | 595.00    | 595.00                     |

**Model 7956 and 7956-SA Gang Programmers** — Similar features to 7228, but program as many as 8 EPROMS at once. 7956-SA stand-alone version copies from a master EPROM. 7956 lab version has all features of stand-alone plus RS-232 interface.

**EPROM:** 2758, 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 2508, 2516, 2532, 2564, 68764, 68766, 5133, 5143. **CMOS:** 27C16, 27C32, 27C64, MC6716. **EPPROM:** 5213, X2816A, 48016, I2816A, 5213H. **MPU (w/adaptor):** 8748, 8748H, 8749, 8749H, 8741, 8742, 8751, 8755.

|                |                                      |               |
|----------------|--------------------------------------|---------------|
| <b>7228</b>    | <b>Advanced Programmer</b>           | <b>\$ 549</b> |
| <b>7128</b>    | <b>Standard Programmer</b>           | <b>429</b>    |
| <b>7956</b>    | <b>Laboratory Gang Programmer</b>    | <b>1099</b>   |
| <b>7956-SA</b> | <b>Stand-Alone Gang Programmer</b>   | <b>879</b>    |
| <b>GDX</b>     | <b>Driver Software</b>               | <b>95</b>     |
| <b>481</b>     | <b>8748 Family Socket Adaptor</b>    | <b>98</b>     |
| <b>511</b>     | <b>8751 Socket Adaptor</b>           | <b>174</b>    |
| <b>755</b>     | <b>8755 Socket Adaptor</b>           | <b>135</b>    |
| <b>CABLE</b>   | <b>RS-232 Cable (specify gender)</b> | <b>30</b>     |

**HEXTRAN Universal HEX File Converter** — Convert assembler output to other formats for downloading to development systems and target boards. Also useful for examining object file, changing load addresses, extracting parts of files. Converts to and from Intel, Motorola, MOS, RCA, Fairchild, Tektronix, TI, Binary and HEX/ASCII Dump formats. For CP/M, CP/M-86, MSDOS, PCDOS .....\$250

Ask about UNIX.

**68000 CROSS-ASSEMBLER** — With exhaustive field testing completed, our 68000 assembler is available for immediate shipment. XMAC68K supports Motorola standard assembly language for the 68000 and 68010. Macros, cross-reference, structured assembly statements, instruction optimization and more. Linker and librarian included. Comprehensive, well-written manual.

To find out more, call us toll-free.

**1-800-448-8500**

(in the U.S. Except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available — please specify. Prices do not include shipping and handling — call for exact quotes. OEM INQUIRIES INVITED.

\*Trademark of Digital Research \*\*Trademark of Microsoft



#### Sales and Development:

10 Summer Street 185-DDJ

P.O. Box 490, Dept.

Rockport, Maine 04856

(207) 236-9055 Telex: 467210 AVOCET CI

#### Corporate Offices:

804 South State Street

Dover, Delaware 19901

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
{  
    case '*': /* Match any following string */  
        for( ; j <= 11; j++)  
            fcb[j] = '?'; /* Pad filetype with trailing */  
        return SUCCESS; /* question marks */  
    case '\0': /* End of fileref */  
        for( ; j <= 11; j++) /* Pad FCB with trailing */  
            fcb[j] = ' '; /* spaces */  
        return SUCCESS;  
    case '.': /* Illegal filetype characters */  
    case ',':  
    case ',':  
    case ',':  
    case ',':  
    case '[':  
    case ']':  
    case '_':  
    case '<':  
    case '>':  
        return ERROR;  
    default:  
        if(c >= '!\' && c <= '~')  
            fcb[j] = c; /* Copy character from fileref */  
        else /* to FCB */  
            return ERROR; /* Nonprintable character or ' ' */  
    }  
}  
  
/* Return ERROR if filetype too long */  
  
return (fref[i] == '\0' ? SUCCESS : ERROR);  
}  
else  
    return ERROR; /* Filename too long */  
}  
  
/* Error report */  
  
error(n,s)  
int n; /* Error code */  
char *s; /* Pointer to optional string */  
{  
    switch(n)  
    {  
        case USER_ERR:  
            printf("\007** ERROR - No user number specified **\n");  
            break;  
        case BAD_REF:  
            printf("\007** ERROR - Illegal file reference: %s **\n",s);  
            break;  
        case BAD_ARGS:  
            printf("\007** ERROR - Illegal command line **\n");  
            break;  
        case BAD_OPT:  
            printf("\007** ERROR - Illegal command line option: ");  
            printf(" %s **\n",s);  
            break;  
        case BAD_USER:  
            printf("\007** ERROR - User number must be inside range");  
            printf(" of 0 to 31 **\n");  
            break;  
        case BAD_DRV:  
            printf("\007** ERROR - Drive names must be inside range");  
            printf(" of 'A' to 'F' **\n");  
    }
```

(Continued on page 58)

# C UTILITY LIBRARY

The C UTILITY LIBRARY is a set of 200+ functions designed specifically for the PC software developer. Use of the Library will speed up your development efforts and improve the quality of your work.

- **BEST SCREEN HANDLING AVAILABLE**
- **WINDOW MANAGEMENT, COLOR GRAPHICS**
- **DOS 2 DIRECTORIES, COMMUNICATIONS**
- **KEYBOARD, PRINTER, TIME/DATE**
- **EXECUTE PROGRAMS, BATCH FILES**
- **STRINGS, BIOS, AND MUCH MORE**
- **ALL SOURCE INCLUDED—NO ROYALTIES**

Available for Microsoft/Lattice \$149, Computer Innovations \$149, Mark Williams \$149, DeSmet \$99. Add \$3 shipping. N.J. residents add 6% sales tax. Visa, MC, checks—10 days to clear. Order direct or through your dealer. Dealer/Distributor inquiries welcome.

**ESSENTIAL SOFTWARE, INC.**  
**(914) 762-6605**  
P.O. Box 1003  
Maplewood, N.J. 07040

Circle no. 36 on reader service card.

## THE LEGENDARY WAY TO SOLVE YOUR BACKUP PROBLEMS

*Here's what Microsystems had to say about our original product: "QBAX will probably become one of those legendary programs that everyone eventually buys. It performs a function useful to anyone with a CP/M system, does it well and quickly, is understandable to the novice computer user."*

*"Every time you run QBAX, the program determines which of your disk files has been changed since the last time it was run. Then it copies these files, and **only** these files, to whatever disk you specify. This is called **incremental backup**, and is the backup method of choice on most large timesharing systems. It will work on any or all active user*

© 1983 by Ziff-Davis Publishing Company

Amanuensis, Inc.  
R.D. #1 Box 236  
Grindstone, PA 15442  
(412) 785-2806

Qbax TM Amanuensis, Inc.  
CP/M Registered TM Digital Research



Circle no. 5 on reader service card.

# At Last! bds C . . . Ver.1.5

**Including a new dynamic debugger  
Still the choice of professionals**

- Compiler option to generate special symbol table for new dynamic debugger by David Kirkland. (With the debugger, the distribution package now requires two disks.)
- Takes full advantage of CP/M\* 2.x, including random-record read, seek relative to file end, user number prefixes, and better error reporting.

**V 1.5 . . . . . \$120.00**  
**V 1.46 . . . . . \$115.00**  
(needs only 1.4 CP/M)

Other C compilers and C related products available . . . Call!

TERMS: CHECK,  
MONEY ORDER, C.O.D.,  
CHARGE CARD  
HOURS: 9 am—5 pm  
Monday —Friday  
**(316) 431-0018**

- Clink option to suppress warm-boot
- New library file search capabilities
- New, fully-indexed 180 page manual
- \* CP/M is a trademark of Digital Research, Inc.

### IT'S HERE! **MONEY MATH**

- Uses BCD internal representation.
- You choose from two types of rounding.
- Configurable exception handling
- Distributed with 12 digits precision. Easily configured for more or less
- Excess 64 exponents

**SOURCE INCLUDED      \$50.00**



Dedicated Micro Systems, Inc.

P.O. Box 481, Chanute, Kansas 66720

include \$2.50 for postage and handling

Circle no. 32 on reader service card.

areas, and so is an absolute **must** for hard- or RAM-disk owners."

Announcing a major enhancement, Qbax2, specifically designed for hard disk users:

■ incremental backup by extent ■ splits files larger than one floppy ■ smart restore: knows exactly which floppies to mount. Can restore individual files or wildcards ■ volume space recovery: prevents consuming floppies endlessly when the same files are backed up repeatedly. ■ time & date stamp & version number on all backup records.

Qbax2 is \$95. For floppy disk users Qbax1 is still available at \$40.

For CP/M 2.2 on 8" SSSD  
& popular 5 1/4" formats  
MC, Visa accepted  
OEM inquiries invited

Shipping:  
\$2 U.S. & Canada, \$4 overseas.

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
break;
case SAME_DRV:
    printf("\007** ERROR - Drive names cannot be equal **\n");
    break;
case OPN_ERR:
    printf("\007\n** ERROR - Cannot open file %s **\n",s);
    reset();
    exit(0);
case READ_ERR:
    printf("\007\n** ERROR - Read error on file %s **\n",s);
    reset();
    exit(0);
case CLS_ERR:
    printf("\007\n** ERROR - Cannot close file %s **\n",s);
    reset();
    exit(0);
case BAD_VFY:
    printf("\007\n** ERROR - Failed verify on file %s **\n",s);
    reset();
    exit(0);
}
printf("\nUsage: BU x[:afn] y [-AFHQSn]\n\n");
printf("      where x = drive name of disk to be backed up\n");
printf("            y = drive name of backup disk\n\n");
printf("      and the optional arguments are:\n\n");
printf("          -A           All files, regardless of");
printf("          status\n");
printf("          -F           Fast copy (without ");
printf("          verification)\n");
printf("          -H           Hard disk (files may be");
printf("          split)\n");
printf("          -Q           Query each file before");
printf("          backup\n");
printf("          -S           System attribute copied to");
printf("          backup\n");
printf("          -n           Backup USER 'n' files only");
printf("          (0-31)\n");
printf("          -afn         Any legal CP/M ambiguous");
printf("          fileref\n");
printf("          (can only be used with -n)");
printf("          option)\n");
exit(0);
}

/* Request a new backup disk to be inserted in the output drive */

new_disk(name,hard_disk)
char *name;
int hard_disk;
{
    char d;

    printf("\007      ** BACKUP DISK FULL **\n\n");
    if(hard_disk)
        printf("WARNING: -H option active - FILE WILL BE SPLIT\n\n");
    printf("Insert new disk into drive %c to continue.\n",name[0]);
    printf("Type 'C' when ready or 'A' to abort ... ");
    while((d = in_chr()) != 'c' && d != 'C' && d != 'a' && d != 'A')
        ;
    if(d == 'a' || d == 'A')
    {
        unlink(name);
        exit(0);
    }
    else
```

(Continued on page 60)

## **APROTEK 1000™ EPROM PROGRAMMER**



only  
**\$250.00**

TECHNICAL  
BREAKTHROUGH  
NOW ALLOWS A  
PRICE  
BREAKTHROUGH

### **A SIMPLE, INEXPENSIVE SOLUTION TO PROGRAMMING EPROMS**

The **APROTEK 1000** can program 5 volt, 25XX series through 2564, 27XX series through 27256 and 68XX devices plus any CMOS versions of the above types. Included with each programmer is a personality module of your choice (others are only \$10.00 ea., when purchased with **APROTEK 1000**). Later, you may require future modules at only \$15.00 ea., postage paid. Available personality modules: PM2716, PM2732, PM2732A, PM2764, PM2764A, PM27128, PM27256, PM2532, PM2564, PM68764 (includes 68766). (Please specify modules by these numbers).

**APROTEK 1000** comes complete with a menu driven BASIC driver programmer listing which allows READ, WRITE, COPY, and VERIFY with Checksum. Easily adapted for use with IBM, Apple, Kaypro, and other microcomputers with a RS-232 port. Also included is a menu driven CPM assembly language driver listing with Z-80 (DART) and 8080 (8251) I/O port examples. Interface is a simple 3-wire RS-232C with a female DB-25 connector. A handshake character is sent by the programmer after programming each byte. The interface is switch selectable at the following 6 baud rates: 300, 1.2k, 2.4k, 4.8k, 9.6k and 19.2k baud. Data format for programming is "absolute code". (i.e., it will program exactly what it is sent starting at EPROM address 0). Other standard downloading formats are easily converted to absolute (object) code.

The **APROTEK 1000** is truly universal. It comes standard at 117 VAC 50/60 Hz and may be internally jumpered for 220-240 VAC 50/60 Hz. FCC verification (CLASS B) has been obtained for the **APROTEK 1000**.

**APROTEK 1000 is covered by a 1 year parts and labor warranty.**

### **FINALLY – A Simple, Inexpensive Solution To Erasing EPROMS**

#### **APROTEK-200™ EPROM ERASER**

Simply insert one or two EPROMS and switch ON. In about 10 minutes, you switch OFF and are ready to reprogram.

**APROTEK-200™ only \$45.00.**

#### **APROTEK-300™ only \$60.00.**

This eraser is identical to **APROTEK-200™** but has a built-in timer so that the ultraviolet lamp automatically turns off in 10 minutes, eliminating any risk of overexposure damage to your EPROMS.

**APROTEK-300™ only \$60.00.**

### **APROPOS TECHNOLOGY**

1071-A Avenida Acaso, Camarillo, CA 93010

CALL OUR TOLL FREE ORDER LINES TODAY:

1-(800) 962-5800 USA or 1-(800) 962-3800 CALIFORNIA

TECHNICAL INFORMATION: 1-(805) 482-3604

Add Shipping Per Item: \$3.00 Cont. U.S.      \$6.00 CAN, Mexico, HI, AK, UPS Blue

Circle no. 8 on reader service card.

# **ConIX™**

## **UNIX™ Technology for CP/M™**

ConIX can provide any 48K+ CP/M-80 system with many advanced capabilities of UNIX. You'll be amazed at what your CP/M micro can do now! ConIX features include:

I/O Redirection and Pipes (uses memory or disk), multiple commands per line, full upper/lower case and argument processing, Auto Screen Paging, Programmable Function Keys, improved User Area Directory manipulation, Command and Extension (Overlay) Path Searching, "Virtual" disk system, 8Mb Print Spooler, extensive preprocessed "Shell" command programming language, 300+ variables, over 100 built-in commands, Math Package, 22 new BDOS SysCalls, Archiver (compacts files for disk space savings of over 50%), On-Line Manual System, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs quickly and easily without any system modifications.

### **The ConIX Operating System**

**List Price: \$165**

Price includes Instructional Manual, 8" SSSD disk and free support. Format conversion available. To order, contact your local dealer, or buy direct and add shipping: \$4.50 UPS, \$10 Canada, \$25 overseas, COD \$2 extra (USA only). NY State residents add sales tax.



**Computer Helper Industries Inc.**  
P.O. Box 680 Parkchester Station, NY 10462

Tel. (212) 652-1786

*Dealer inquiries invited!*

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Comp. Helper Ind. Inc.

Circle no. 22 on reader service card.

**Super Fast**

# **Get Fast Relief!**

**S-100!**

**IBM PC/XT!**

**TRS\*80 II!**

**EPSON QX10!**

**ZENITH Z-100!**

If you've been "patient" with slow disk drives for too long, SemiDisk will relieve your suffering.

### **Fast-acting.**

The SemiDisk, a super-fast disk emulator, stores and retrieves data much faster than either a floppy or hard disk.

### **Easy to apply.**

Installation is as easy as plugging the SemiDisk into an empty slot of your computer, and running the installation software provided.

### **Regular and extra-strength.**

SemiDisk I is the standard model for S-100, SemiDisk II offers extra speed and flexibility for custom

S-100 applications.

### **Contains gentle buffers.**

CP/M®80 installation software includes SemiSpool, which buffers print data in the SemiDisk. This allows the computer to be ready for other uses immediately after issuing a print command.

### **No emulator amnesia.**

The optional Battery Backup Unit (BBU) plugs into the SemiDisk, and supplies power even when the computer is off. A battery keeps the data alive during power outages of four hours or more.

### **Stops head-aches.**

Unlike a hard disk, which can 'crash' its head on the rotating disk

surface, and a floppy, which grinds the disk constantly, the SemiDisk gives you ultra-fast, silent data transfer.

And SemiDisk's price won't raise your blood pressure.

### **512K 1Mbyte**

|                      |        |        |
|----------------------|--------|--------|
| SemiDisk I, S-100    | \$995  | \$1795 |
| SemiDisk II, S-100   | \$1245 | \$2095 |
| SemiDisk, TRS-80 II  | \$995  | \$1795 |
| SemiDisk, IBM PC     | \$945  | \$1795 |
| SemiDisk, Epson QX10 | \$995  |        |
| SemiDisk, BBU        | \$150  |        |

# **SEMDISK**

**SemDisk Systems, Inc.**  
P.O. Box GG,  
Beaverton, Oregon 97075  
503-642-3100

Call 503-646-5510 for CBBS, NW and 503-775-4838 for CBBS/PCS, both SemDisk-equipped computer bulletin boards (300 1200 baud). SemDisk, SemiSpool trademarks of SemDisk Systems. CP/M trademark of Digital Research.

Circle no. 85 on reader service card.

# Archiving Files Listing

(Listing Continued, text begins on page 36)

```
    printf("\n\n");
    bdos(RESET_DRV,NULL);           /* Reset drives */
}

/* Reset user and drive codes to entry values */

unset()
{
    bdos(USER_CODE,ent_user);
    bdos(SEL_DRV,ent_drv);
}

/* Get character from current CP/M CON: device without echo */

in_chr()
{
    int c;

    do
        c = bdos(DIR_IO,0xff);
    while(!c);
    return c;
}

/* Additional function required for DeSmet C version of BU86.C */

#if DESMET
bdos(beta,delta)
int beta,
    delta;
{
    return _os(beta,delta);
}
#endif

/** EXPLANATION OF AZTEC CII STDIO.H FUNCTIONS **/

/* bdos(bc,de)  ! DeSmet equivalent is defined under FUNCTIONS !
 * int bc,de;
 *
 * Calls CP/M's BDOS with 8080 CPU register pair BC set to "bc"
 * and register pair DE set to "de". The value returned by the
 * 8080 CPU accumulator is the return value.
 *
 * movmem(src,dest,length)  ! DeSmet equivalent is defined  !
 * char *dest, *src;          ! under DEFINITIONS          !
 * int length;
 *
 * Moves data from "src" to "dest". The number of bytes is
 * specified by the parameter "length".
 *
 * strcmp(str1,str2,max)  ! Identical for DeSmet !
 * char *str1,*str2;
 * int max;
 *
 * Compares "str1" to "str2" for at most "max" characters, and
 * returns NULL if strings are equal, -1 if "str1" is less than
 * "str2", and +1 if "str1" is greater than "str2".
 */
/* End of BU.C */
```

End Listing

## WIZARD C

Fast compiles, fast code and great diagnostics make Wizard C unbeatable on MSDOS. Discover the powers of Wizard C:

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

The new standard for C Compilers on MSDOS!

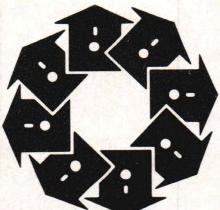
Only \$450

For more information call (617) 641-2379  
Wizard Systems Software, Inc.  
11 Willow Ct., Arlington, MA 02174  
Visa/Mastercard accepted

**WSS**

Circle no. 116 on reader service card.

**UNPARALLELED  
PERFORMANCE  
and PORTABILITY  
in an ISAM PACKAGE  
at an UNBEATABLE  
PRICE**



**c-tree™**  
BY FAIRCOM

2606 Johnson Drive  
Columbia MO 65203

## The Tools You Need To C You Thru.

Now the *WizardWare™*  
Applications Programmers Toolkit  
provides everything you need  
to increase your C programming productivity.

### APT™ features:

- File Handlers
  - Direct Access
  - Keyed Access
- Generic Terminal Driver
- String Handling
  - Manual On Disk (45 pages)
  - Tutorial On Disk (33 pages)
  - Detailed Brochure on request
- String Math
- Screen Generator
- Report Generator
- FIFO Que Routines
- Source Code

UNIX™ System V Version (available 1-1-85) . \$995  
MS-DOS™ Version . . . . . \$495  
BDS C™ Version . . . . . \$395  
Manual Only\* . . . . . \$50

\*Manual Cost will be applied to APT™ purchase price within 30 days (\$10 re-stocking charge). U.S. funds only, please.

Trademark owners: UNIX™ (AT&T Bell Labs), MS-DOS™ (Microsoft, Inc.), BDS C™ (BD Software), Wizardware™ and APT™ (Shaw American Technologies)

Call: (800) 443-0100 Ext. 282

Ask for APT™ or Send Check To:  
**Shaw ★ American Technologies**

*WizardWare™*

830 South Second Street - Box 648

Louisville, KY 40201, U.S.A.

(C.O.D. and Foreign Orders - Add \$5 Shipping/Handling)

References: Bank of Louisville, Citizens Fidelity Bank, Louisville Chamber of Commerce



Circle no. 86 on reader service card.

The company that introduced micros to B-Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B-Tree based file handlers. With c-tree™ you get:

- complete C source code written to K & R standards of portability
- high level, multi-key ISAM routines and low level B-Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

**\$395 COMPLETE**

Specify format:  
8" CP/M® 5 1/4" PC-DOS 8" RT-11

for VISA, MC or COD orders, call  
1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc.  
c-tree and the circular disc logo are trademarks of FairCom

© 1984 FairCom

Circle no. 37 on reader service card.

# MBOOT and MODEM7 for the C-64's CP/M

by Walt Piotrowski

If you have CP/M for your Commodore 64, you may have noticed that the ads for CP/M software never mention that the program is available in Commodore's disk format. If you have tried to buy any of this software, you have found out why the ads don't mention it: with one exception, diskettes for the C-64 are not available (the only exception that I am aware of is Turbo Pascal from Borland International).

One company, Laboratory Microsystems, tried quite hard to help me buy their product but simply did not have access to any equipment that could write a diskette for the C-64. One of their suggestions was to provide the software on an IBM PC-compatible diskette; since I have access to a PC, I could transfer the program to the C-64. Unfortunately, I did not have any software for my C-64 that could do a download either.

## **MBOOT on the C-64—Download Only**

MBOOT, from the *DDJ* CP/M exchange column, was a good attempt at a generic program to download files using Ward Christensen's Xmodem file transfer protocol. In principle, all you have to do is take the listing, change a few equates to match your modem, type in the program, and assemble it, and you are ready to download files. If you are familiar with the way that CP/M is implemented on the C-64 (see the references at the end of the article if you aren't), you have probably guessed that modifying MBOOT for this computer isn't quite that easy.

Normal CP/M operation on the C-64 does not require access to the RS-232 port, so there is no code in the BIOS to handle this port. Because the modem is controlled through the RS-232 port, the modification for MBOOT required some 6510 code in addition to

***"This article is about the MBOOT download program (DDJ, July 1982) and the modifications that were necessary to make it work on the C-64."***

This article is about the MBOOT download program (*DDJ*, July 1982) and the modifications that were necessary to make it work on the C-64. Once you have MBOOT working, you can use a modem to download public domain software from RCPMs, and, assuming you have access to a machine with one of the more popular formats, you can also use it to set some of the commercial software that you need.

*Walt Piotrowski, State University of New York, Binghamton, NY 13901.*

MBOOT's 8080 code. This 6510 code makes RS-232 I/O requests to the Commodore I/O kernel. The problem is made a little more complex by the fact that the kernel handles the RS-232 transfers one bit at a time by servicing nonmaskable interrupts from a 6526 Interface Adapter chip. Normally, the kernel accepts an RS-232 request and sets up the hardware for the transfer but returns control to the requestor before the actual transfer is complete. The kernel then services the interrupts as they come along; other 6510 activities proceed in parallel with

the RS-232 transfer.

Under CP/M, however, it is quite normal to shut off the 6510 for long periods while programs run in the Z80. But the hardware does not allow the Z80 to receive these interrupts, nor does it contain any provision for switching on the 6510 if an interrupt arrives. This means that, unless MBOOT contains some provision to ensure that the 6510 is running (almost) continuously, RS-232 transfers will become garbled. Making sure that the 6510 is running while a character is being sent is not a big problem since transmission is totally under MBOOT's control. The 6510 code that I have added to make the transmission request to the I/O kernel includes a loop that waits in the 6510 until the kernel's RS-232 status byte shows that the character has been completely transmitted. Data reception is a little more complex.

Listing One (MBOOT64, page 67) is the modification of MBOOT's original 8080 code. If you were to compare MBOOT64 to MBOOT, you would find that MBOOT's inline I/O instructions, which originally were directed to the modem port, have been replaced with calls to a set of 6510 interface routines. These routines can be found at the end of Listing One. They allow the program to open the RS-232 channel (OPMDEM), close it (CLMDEM), send a character to the modem (WRMDEM), receive a character (RDMDEM), receive a character in a time out loop (RDMDEM2), and read a keyboard character (KYBD). The interface routines do not actually perform these functions: they set up parameters in memory that eventually cause a transfer to corresponding 6510 routines that do the work required. Listing Two (CPMMD65, page 79) contains the 6510 counterparts of each interface routine.

The actual transfer of control between the Z80 and the 6510 takes place at the label GO6510 in the interface routine section of MBOOT64. The transfer of control employs a user function in the BIOS that Commodore has provided to standardize the interface between CP/M programs and user-developed 6510 programs. This user function and the standard interface are described in the Commodore 64 CP/M user's guide. The modifica-

tions to MBOOT make use of request code 9 in this standard interface, and the program should be usable even if future modifications are made to Commodore's CP/M.

MBOOT (and MBOOT64) has two modes of operation: a file receive mode and a dumb terminal mode. In the file receive mode, the XMODEM protocol provides a strict handshake sequence between the computer transmitting the file and the computer receiving the file. Because of this handshaking, data reception in the file receive mode is not much more difficult than data transmission: MBOOT (and MBOOT64) sends a character to the computer that is transmitting the file then enters a wait loop. This transmitted character (ASCII ACK or NAK) is a signal that the receiving computer expects one sector of data.

In the original MBOOT, the wait loop contained code to poll the modem and a timeout counter to prevent a permanent hangup if the data never arrived. A vestige of this loop is at the label MWTI in MBOOT64. At MWTI (location 03E5), you will find a call to RDMDEM2, which activates the "read with wait" function in the 6510; this read with wait is at label INPUT2 (line 139) in CPMMD65.

Immediately after a byte is received, control is returned from the 6510 to the Z80. At this point, the 6510 is shut off while the Z80 takes care of the received character. However, this time period is very short, and the 6510 is always back on in time to accurately receive the next character.

In the dumb terminal mode, characters arrive at random times, making it impossible to predict the exact moment when a character will begin to arrive. The modification to this mode takes advantage of the fact that the terminal code is a very short loop in which a modem status check occurs every few microseconds. The modification simply puts this status check function into the 6510. Because the 6510 will now be turned on every few microseconds, it will be able to service the RS-232 interrupt either as soon as it arrives or very shortly thereafter.

The start of the terminal loop is at label TERM (location 019A) in MBOOT64. In this loop, a call to RDMDEM, one of the 6510 interface

# DeSmet C

**8086/8088 Development Package \$109**

## FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete STDIO Library (>120 Func)

## Automatic DOS 1.X/2 X SUPPORT

## BOTH 8087 AND S/W FLOATING POINT OVERLAYS

## OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

**SYMBOLIC DEBUGGER \$50**

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

**DOS LINK SUPPORT \$35**

- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check:  Dev. Pkg (109)  
 Debugger (50)  
 DOS Link Supt. (35)

SHIP TO: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

**C W A R E**  
CORPORATION

P.O. BOX C  
Sunnyvale, CA 94087  
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on US Bank and in US Dollars. Call 9 a.m. - 1 p.m. to CHARGE by VISA/MC/AMEX.

Circle no. 18 on reader service card.

routines, activates the modem status check function in the 6510. This function is located at label INPUT (line 110) in CPMM65.

Each time INPUT is entered, it examines the kernel's status byte to see if the kernel has begun to input a character. If it has, the routine waits until the character has been completely transferred (Listing Two, lines 126–131). INPUT and RDMDEM also return the modem character to the caller if one is ready. The wait loop in INPUT, plus the one in the character transmit routine (Listing Two, lines 102–104), keeps the 6510 running a very high percentage of the time, and full duplex operation at 300 baud proceeds smoothly. (More information on data rates is included near the end of the article.)

There is an additional complicating factor involving the terminal mode. As you probably know, the C-64's native character set is not ASCII. When Commodore implemented CP/M, it chose to generate ASCII with a new keyboard scan routine. This CP/M keyboard

scan runs partly in the 6510 and partly in the Z80. Unfortunately, the key debounce loop is located in the Z80. This means that whenever a key is pressed the Z80 enters this delay loop; if a modem character happens to come along while the Z80 is in this loop, the modem character gets lost or garbled.

This turns out to be a significant problem because in the terminal mode every character typed at the C-64 is echoed by the computer at the other end of the line. At normal typing speeds, this debounce delay loop causes almost every echoed character to be garbled—it becomes impossible to read what you have typed. The solution was to add a new keyboard routine in the 6510 and to use that routine instead of the CP/M keyboard routine while in the terminal loop. The interface between MBOOT64 and the new keyboard routine is at label KYBD (location 04CC) in MBOOT64, and the corresponding 6510 routine is at label KBDCHR (line 154) in CPMM65.

The keyboard routine that I imple-

mented is a very simple one: it uses the normal C-64 kernel keyboard routines (not the CP/M routine) and simply translates received characters from the C-64 character set into ASCII through a translation table. This routine is adequate for the terminal mode, at least for me, but you will notice that it does not allow you to switch to an all uppercase keyboard mode. You will also notice that the characters with ASCII codes 32 (space) and below have an auto repeat. This repeat is built into the native C-64, and it was easier to leave it in than to try to undo it.

The modification to MBOOT includes one further change. The original MBOOT contained code to buffer 16 sectors before actually doing a disk write. I tried it both ways and, with 1541 disk drives, found no speed advantage in doing this buffering. Because the program is easier to type in without this code, I omitted it. (Incidentally, the comments in the MBOOT64 listing were not in the original DDJ column, but I needed them to help me to understand the program. I believe that they are correct, but it isn't easy to debug a comment, so they may contain some misinterpretations.)

The Figure (at left) shows where the 8080 and 6510 portions of the modified MBOOT reside in memory. The 8080 code begins at the beginning of the TPA (\$100 in Z80 space) while the 6510 code begins at location \$600. You will note from Listing Two that the 6510 code is actually assembled at \$1600 due to the \$1000 offset between the address spaces of the two processors. The data shared between the two parts of each function resides in memory locations \$603–\$605 in Z80 space.

The figure also shows the location of the character translation table that is used in the terminal keyboard routine and the new location of the kernel's RS-232 buffers. These RS-232 buffers originally were moved from their normal location so that a 48K CP/M could be used. Since writing the program, I have discovered that the 48K CP/M will occasionally send a garbage character out the RS-232 port while doing disk transfers. Most RCPMs ignore these characters, but they cause the XMODEM program on a few RCPMs to abort while doing file transfers. I have found that it is best to use a 44K CP/M

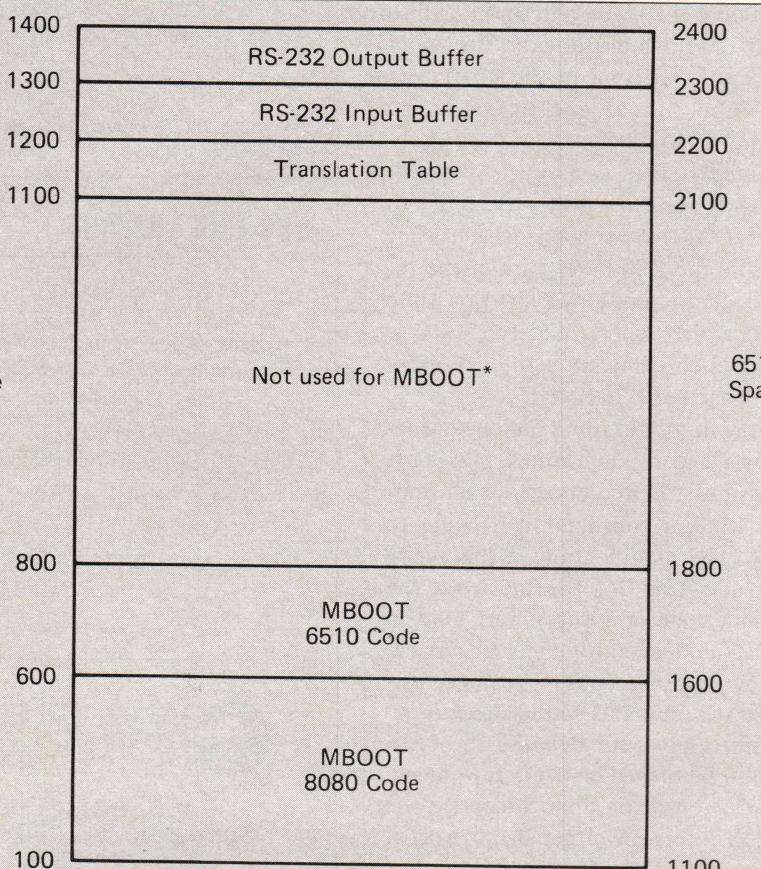


Figure 1

Memory Layout for C-64 MBOOT

\*This space was left for compatibility with MODEM464

# DSD 80

## The Dynamic Screen Debugger for CP/M

DSD-80 is the most advanced debugging program available for CP/M-80 and compatible operating systems. Spend less time debugging and more time programming because DSD is the only utility to offer all of the most important features in one package.

### Satisfaction is guaranteed or a full refund!

Full featured simulator provides for:  
Execution only within boundaries  
Write protected memory  
Stack overflow protection  
Stack underflow protection  
Real time subroutines  
Breakpoint on specified memory or register values  
Full screen display includes:  
2 memory displays  
Register display  
Stack display  
Instruction display  
Fully upward compatible with DDT

Single keystroke commands for:  
Stepping instructions  
Subroutine execution  
Scrolling memory displays  
Full symbol support including:  
Loading symbol files  
Interactive symbol definitions  
Endorsed by Leo Zolman — author of BDS C  
Uses less than 15k of memory  
On line help information  
Clearly written 50 page manual  
Free updates for one year

**Only \$195.00**

## SOFT ADVANCES

P.O. Box 49473 Austin, TX 78765 (512) 478-4763

Available on 8" SSSD & various 5.25" formats  
IBM PC version available soon!  
Visa & MC accepted

Circle no. 63 on reader service card.

## Powerful in circuit emulation, priced well within your grasp. That's NICE.™

NICE may be only 3" square and 1/2" thick, but it hands you full speed, real-time emulation—over 50 emulation functions, software breakpoints, all memory addresses and all I/O ports.

Just plug NICE directly into the target MP socket and any RS232 terminal for system development, troubleshooting, debugging or testing... at home, in the lab or in the field.

And NICE hands you all this performance, portability and versatility for only \$498... the best emulator price/performance ratio on the market, hands down.

Call in your order today using

your VISA or Mastercard number: (800) NICOLET outside CA, or (415) 490-8300 in CA.

Or send your

check or

money

order

to NICE,

Nicolet

Paratronics

Corporation,

201 Fourier

Avenue,

Fremont, CA 94539.

\*Payment by check, money

order, VISA or MasterCard.

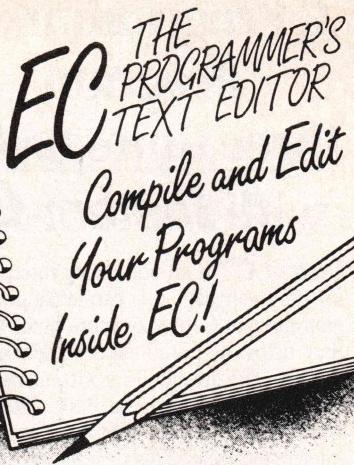
NICE is a trademark of Nicolet

Paratronics Corporation

\*Z80 is a trademark of Zilog, Inc.

Nicolet  
NICE  
For the  
Z80  
NOW AVAILABLE—NICE for the Z80  
COMING: NICE 8085—Sept. '84  
NICE NSC800—Oct. '84  
MORE TO COME!!

Circle no. 60 on reader service card.



Now you can edit, compile and test your program from inside the EC editor.\* In fact, EC gives you complete access to the operating system. Do a directory, copy or delete files, even run other programs without ever leaving EC!

### MULTIPLE WINDOWS

Forget about dumping a file you're editing just so you can see what is in another file... open a new window, up to five of them, and read in the file you want to use.

All windows can be shown on the screen at the same time; or the screen can be dedicated to just a single file, while the others are kept in the background—only a keystroke away. You can even cut and paste blocks of text between windows!

### FULL SCREEN EDITOR FOR THE IBM

Developed specifically for the IBM PC, EC makes extensive use of the entire PC keyboard so editing is fast and intuitive.

In addition to standard editing features, EC supports command and text macros, word wrap, paragraph reformatting, horizontal scrolling, and control for color monitors.

### DEMO DISK IS ONLY \$5

Call or write for our full-featured demo. You'll get a standard version of EC that handles up to 10K worth of files and a complete set of documentation.

See for yourself how pleasant it is to use an editor that gives you both unrestricted access to the operating system and multiple windows.

You won't be disappointed!

**EC EDITOR - \$125**  
**EC DEMO - \$5 (plus \$1.65 for COD)**

\* EC runs on an IBM PC, or look-alike, with at least 128K RAM under DOS 1.1 or higher. To use the DOS interface feature you must have DOS 2.0 or higher and enough RAM to run the additional program.

IBM is a trademark of International Business Machines  
MO. RESIDENTS ADD 6% SALES TAX

**SOURCE**

C SOURCE

12801 Frost Road • Kansas City, MO 64138

**816-353-8808**

Circle no. 15 on reader service card.

# C Programmers: Program three times faster with Instant-C™

**Instant-C™** is an optimizing interpreter for the C language that can make programming in C three or more times faster than using old-fashioned compilers and loaders. The interpreter environment makes C as easy to use as Basic. Yet **Instant-C™** is 20 to 50 times faster than interpreted Basic. This new interactive development environment gives you:

**Instant Editing.** The full-screen editor is built into **Instant-C™** for immediate use. You don't wait for a separate editor program to start up.

**Instant Error Correction.** You can check syntax in the editor. Each error message is displayed on the screen with the cursor set to the trouble spot, ready for your correction. Errors are reported clearly, by the editor, and only one at a time.

**Instant Execution.** **Instant-C™** uses no assembler or loader. You can execute your program as soon as you finish editing.

**Instant Testing.** You can immediately execute any C statement or function, set variables, or evaluate expressions. Your results are displayed automatically.

**Instant Symbolic Debugging.** Watch execution by single statement stepping. Debugging features are built-in; you don't need to recompile or reload using special options.

**Instant Loading.** Directly generates .EXE or .CMD files at your request to create stand-alone versions of your programs.

**Instant Floating Point.** Uses 8087\* co-processor if present.

**Instant Compatibility.** Follows K & R standards. Comprehensive standard library provided, with source code.

**Instant Satisfaction.** Get more done, faster, with better results. **Instant-C™** is available now, and works under PC-DOS, MS-DOS\*, and CP/M-86\*.

Find out how **Instant-C™** is changing the way that programming is done.

**Instant-C™** is \$500. Call or write for more information.

**Rational**  
Systems, Inc.

(617) 653-6194

P.O. Box 480

Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.) 8087 (Intel Corp.), CP/M-86 (Digital Research, Inc.). Instant-C (Rational Systems, Inc.)

when using MBOOT or other modem programs, even though they will work most of the time with a 48K CP/M.

There are two ways to add the 6510 code to MBOOT. The easiest, not that it works, is simply to include the code at the end of the 8080 assembly as a set of hex constants. (Don't forget that it must begin at location \$600.) The second way, if you have a 6510 assembler that runs on the native 6510, is to assemble the program and load the object code into memory prior to doing a CP/M cold start; the cold start leaves these 6510 program locations unaffected. Once CP/M has been loaded, use DDT to load in MBOOT's 8080 COM file and save the combined program with "SAVE 7 C64MBT.COM."

## MODEM4—Upload and Download

After I had MBOOT working, I began to run up my long distance phone bill calling RCPMs and looking for a copy of the source code for MODEM7. During the search, I found many RCPMs with copies of the object code but none with the source code. Since this search was getting rather expensive, I decided to stop and go to work on a copy of MODEM4, a program I had found and downloaded while looking for MODEM7.

MODEM4 apparently is a modem program that was part of the evolution toward MODEM7. You can use it as a terminal program as well as to upload and download files using Christensen's protocol. I modified it for the C-64 and used it to put copies of itself on several RCPMs in the Northeast. It's called MODEM464 and a short .DOC file accompanies it.

To find it, look for MODEM464.OBJ. Download it with MBOOT, rename it MODEM464.COM, and run it. If you run it by typing MODEM464 T, you will be in the terminal mode and the program will act the same as MBOOT. When you are ready to download a file (after you have set up the RCPM's XMODEM), exit from the program with CTL-E and type "MODEM464 R filename.ext." This will reload the program from disk and start it into file receive mode. Once the file has been transferred, the program goes directly back into terminal mode. If you want to send a file, type "MODEM464 S filename.ext." The program will send the file

and go to terminal mode when completed, just as it does in the file receive mode.

## MODEM7—Upload, Download, and More

A short time after I finished the work on MODEM4, I came across the source code for MDM730, a version of MODEM7. MODEM7 is a full-feature modem program with a terminal mode, a file upload and download capability, a terminal mode capture buffer for ASCII text, and a terminal mode ASCII file transmit function. The ASCII capture buffer is useful for saving text, such as the directory of an RCPM, that you don't want to lose but can't transfer in file (XMODEM) form. The ASCII file transmit function is useful if you have a long message that you want to leave on a bulletin board without using the time (and money) to compose it while on-line. These ASCII file features are also useful if you wish to exchange text files with someone who does not have a program with the XMODEM protocol. MDM730 will also autodial with several different kinds of modems, and it allows you to echo text directly to your printer. (The modified version will autodial only with a Hayes modem.)

The modified program, called MD730C64, is harder to learn to use than either MBOOT or MODEM4 because it has so many features. To get going, you must download three files: the executable program (MD730C64-OBJ), the documentation for the original program (MDM730.DOC), and the notes that explain the few differences between MDM730 and MD730C64 (MD730C64.DOC).

Getting MD730C64.OBJ is easy: download it using MBOOT, rename it to a .COM file, and you are ready to run it. Getting the documentation, however, is a bit more difficult. Most RCPMs store long text files in a squeezed format to conserve disk space and to shorten the download time; you can recognize squeezed files because they have a Q in the file extension. Therefore, you may find MD730C64-DQC instead of MD730.DOC. If you do, you must also download a program to unsqueeze the file on your own system. This unsqueeze program has many versions, but it almost always has a name that starts with USQ.

Getting the original MDM730 program documentation file may be tougher still. On most RCPMs you must download a library file called MDM730.LBR. A library file is a collection of individual files relating to the same program—MDM730.DQC is one of the individual files in MDM730.LBR. To extract the file that you want from a library, you need a library utility program called LU (and its documentation LU.DQC). Once you have learned to use LU, you can extract MDM730.DQC from the MDM730 library file. This library also contains a program that lets you change the list of RCP/M phone numbers in MDM730 to ones that you call more frequently.

Don't be discouraged by the complexity of this whole process. Once you start to find your way around RCPMs, you will find that most of the really worthwhile programs are in libraries and have squeezed documentation. Your experiences and the utility programs that you have downloaded will pay off later.

I ran into some interesting problems when modifying MDM730. The biggest was the size of the source code. The squeezed file that I downloaded was 105K, which will fit on a C-64 diskette. When unsqueezed, it becomes 158K, which will not. To get an assembly listing, I had to use my modified MO-

DEM4 to transfer the squeezed source code and a copy of an unsqueeze program to a Cromemco Z80 system at the university where I work. The disks on this system hold 250K, and I was able to unsqueeze the file and assemble it there. My original intent was to modify MDM730 by editing and reassembling the original source file, but I decided that editing and assembling on one machine and testing on another 30 miles away would be a nearly impossible task. Because of that, I chose to make the modification in the form of a patch overlay.

### Caveats

MDM730 is a very complex program; consequently, so is MD730C64. I have tested it carefully, but you may uncover some minor problems. Also, because of equipment limitations, I have not been able to test any of these programs at data rates higher than 300 baud. Because of the switch that takes place between processors while the data is arriving, the program may need more work to run successfully at higher rates. Please let me know if you uncover any problems and if you are able (or unable) to run at rates higher than 300 baud.

A note about modems: When I began accessing RCPMs, I used the VIC modem that I had been using for CompuServe. I discovered that, while it works

with CompuServe because they have a local phone number, on the typical long distance call with a noisy line the VIC modem simply doesn't work well. If you plan to call RCPMs, you will need a modem with better immunity to noise.

### References

1. Commodore Business Machines. *Commodore 64 CP/M Operating System User's Guide*. Indianapolis: Howard W. Sams and Co., 1983.
2. Head, Gene. "CP/M Exchange." *Dr. Dobb's Journal*, July 1982, pages 42–50.
3. Hoff, Irvin M. "MDM730 (notes on how to use)." File MDM730.DOC, 1984.
4. Piotrowski, Walt. "CP/M on the Commodore 64." *Dr. Dobb's Journal*, June 1984.

### RCPMs with MD730C64

1. Allentown (PA) RBBS/RCPM, Bill Earnest, (215) 398-3937.
2. Bearsville (NY) Town SJBBS, Hank Szyszka, (914) 679-6559.
3. St. Mary's College (MD) RCP/M, Jonathan Crawford and Bob Beasley, (301) 863-7165.

Thanks to the SYSOPS for permission to use their names and numbers. **DDJ**

### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 194.

## MBOOT and MODEM7 (Text begins on page 62)

### *Listing One*

B\$MBOOT64.PRN

\*\*\*\*\*

; MBOOT64 - FILE DOWNLOAD PROGRAM

ORIGINAL MBOOT BY GENE HEAD  
D.B.J. OCTOBER, 1982

MODIFIED FOR THE C-64 BY  
WALT PIOTROWSKI

\*\*\*\*\*

|        |         |     |             |                                     |
|--------|---------|-----|-------------|-------------------------------------|
| 0000 = | BASE    | EQU | 0           |                                     |
| 0005 = | EXITCHR | EQU | 05H         | ;CTL-E TO EXIT TERM MODE TO CP/M    |
| 0004 = | FILCHR  | EQU | 04H         | ;CTL-D TO RECEIVE FILE              |
|        |         |     |             |                                     |
| 0600 = | MD65Z   | EQU | 600H        | ;LOC OF 6510 MODEM RTNE (Z80 SPACE) |
| 1600 = | MD65    | EQU | MD65Z+1000H | ; (6510 SPACE)                      |
| 0603 = | MDFUNC  | EQU | MD65Z+3     | ;MODEM FUNCTION CODE                |
| 0604 = | MDCHAR  | EQU | MD65Z+4     | ;MODEM CHARACTER                    |

(Continued on next page)

## Listing One

```

0605 =      MOREC EQU     MD65Z+5 ;MODEM CHAR RCVD FLAG
;
F900 =      BIOSFN EQU     0F900H ;BIOS65 FUNCTION CODE
F906 =      BIOSAD EQU     0F906H ;MODEM ROUTINE INDIRECT ADDRESS
CE00 =      ON6510 EQU     0CE00H ;MEM LOC TO TURN 6510 ON
;
000A =      ERRLIM EQU     10    ;NUMBER OF TRIES FOR ONE BLOCK
;
0001 =      SOH    EQU     1      ;ASCII CONTROLS
0004 =      EOT    EQU     4
0006 =      ACK    EQU     6
0015 =      NAK    EQU     15H
0018 =      CAN    EQU     18H
000A =      LF     EQU     10
000D =      CR     EQU     13
;
0005 =      BDOS   EQU     BASE+5
005C =      FCB    EQU     BASE+5CH
;
0100          ORG     BASE+100H
;
0100 210000  LXI     H,0      ;CLEAR HL
0103 39       DAD     SP       ;MAKE A COPY OF SP
0104 222B05  SHLD    STCK    ;SAVE FOR EXIT
0107 312B05  LXI     SP,STCK ;POINT TO LOCAL STACK
010A CDF903  CALL    INITADR ;SET UP BIOS CALLS
010D CD3604  CALL    ILPRT   'C-64 MBOOT AS OF '
0110 432D363420 DB      '4/1/84',CR,LF,0
0121 342F312F38 DB      FCB+1  ;LOOK AT FILE NAME
012A 3A5D00  LDA     CPI     ' ' ;BLANK?
012D FE20  CPI     JNZ     TERM1  ;NO - GO PROCESS
012F C25501  JNZ     ILPRT   ;WRITE ERROR MSG
0132 CD3604  CALL    ILPRT   ' '
;
0135 2B2B4E4F20 DB      '++NO FILE NAME SPECIFIED++',CR,LF,0
0152 C37304  JMP     EXIT   ;
;
0155 CD3604  TERM1  CALL    ILPRT   ;WRITE FIRST MSG
0158 0D0A544552 DB      CR,LF,'TERMINAL MODE',CR,LF
0169 43544C2D45 DB      'CTL-E EXITS TO CP/M'
017C 0D0A  DB      CR,LF
017E 43544C2D44 DB      'CTL-D STARTS FILE XFER'
0194 0D0A00  DB      CR,LF,0
0197 CD7E04  CALL    OPMDEM ;OPEN FOR MODEM
;
; DUMB TERMINAL LOOP
;
019A C0CC04  TERM   CALL    KYBD   ;CHECK FOR KYBD CHAR
019D FE00  CPI     0       ;CHAR RECEIVED?
019F CAAF01  JZ     TERML  ;GO CHECK FOR MODEM CHAR
01A2 FE05  CPI     EXITCHR ;EXIT TO CPM?
01A4 CA7304  JZ     EXIT   ;
01A7 FE04  CPI     FILCHR ;BEGIN RECEIVING FILE?
01A9 CABF01  JZ     RCVFIL ;DISPLAY RECVD CHAR
01AC CD9004  CALL    WRMDEM ;OUTPUT KYBD CHAR TO MODEM
;
01AF CD9E04  TERML  CALL    RDMDEM ;INPUT MODEM CHAR
01B2 FE00  CPI     0       ;0 - NO CHAR RECVD
01B4 CA9A01  JZ     TERM   ;BACK TO MAIN LOOP
01B7 E67F  ANI     7FH    ;REMOVE PARITY
01B9 CD1204  CALL    TYPE   ;DISPLAY RECVD CHAR
01BC C39A01  JMP     TERM   ;BACK TO MAIN LOOP
;
; FILE RECEIVE
;
01BF CD1403  RCVFIL CALL    ERASFIL ;SEE IF FILE EXISTS
01C2 CD5B03  CALL    MAKEFIL ;OPENFILE

```

(Continued on page 70)

# (LISP)

**Artificial Intelligence Language**  
**UO-LISP Programming Environment**  
**The Powerful Implementation of LISP**  
**for MICRO COMPUTERS**



**LEARN LISP System (LLS.1)** \$39.95  
 (see description below)

**UO-LISP Programming Environment**  
**Base Line System (BLS.1)** \$49.95

**Includes:** Interpreter, Compiler, Structure Editor, Extended Numbers, Trace, Pretty Print, various Utilities, and Manual with Usage Examples. (BLS.1) expands to support full system and products described below.

**UO-LISP Programming Environment:** The Usual LISP Interpreter Functions, Data Types and Extensions, Structure & Screen Editors, **Compiler, Optimizer, LISP & Assembly Code Intermixing**, Compiled Code Library Loader, I/O Support, Macros, Debug Tools, Sort & Merge, On-Line Help, Other Utility Packages, Hardware and Operating System Access, Session Freeze and Restart, Manual with Examples expands to over 350 pages. Other UO-LISP products include: LISPTEX text formatter, LITTLE META translator writing system, RLISP high level language, NLARGE algebra system. Prices vary with configurations beyond (BLS.1) please send for *FREE catalog*.

**LEARN LISP System (LLS.1):** Complete with LISP Tutorial Guide, Editor Tutorial Guide, System Manual with Examples, Full LISP Interpreter, On-Line Help and other Utilities. LEARN LISP fundamentals and programming techniques rapidly and effectively. This system does not permit expansion to include the compiler and other products listed above.

**LISP Tutorial Support (LTS.1):** Includes LISP and Structure Editor Tutorial Guides, On-line Help, and History Loop. This option adds a valuable learning tool to the UO-LISP Programming Environment (BLS.1). Order (LTS.1) for \$19.95.

**REQUIRES:** UO-LISP Products run on most Z80 computers with CP/M, TRSDOS or TRSDOS compatible operating systems. The 8086 version available soon.

**TO ORDER:** Send Name, Address, Phone No., Computer Type, Disk Format Type, Package Price, 6 1/2% Tax (CA residents only), Ship & Handle fee of \$3.00 inside U.S. & CN, \$10 outside U.S., Check, Money Order, VISA and MasterCard accepted. With Credit Card include exp. date. Other configurations and products are ordered thru our *FREE catalog*.

**Northwest Computer Algorithms**  
 P.O. Box 90995, Long Beach, CA 90809 (213) 426-1893

Circle no. 61 on reader service card.

Rollinsoft

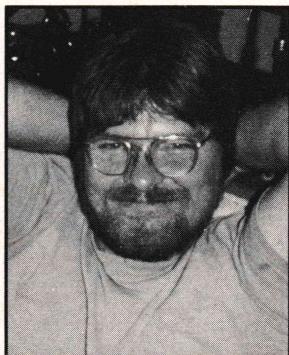
PCsoftware

Don Buresh

Ron Watson

## PC FIRING LINE / PC UNDERGROUND™

Technical Disk Magazine

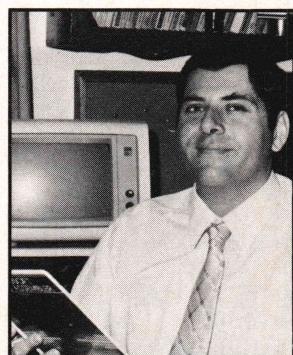


Dan Rollins

Throw away your paper magazines! Save shelf space, save programming time, with the sizzling ready-to-use source code provided in each powerful bimonthly issue (currently 2 DS/DD disks per issue).

Dan Rollins, captured here relaxing after installing a device driver, has provided PCFL/PCUG with Polymaze, a maze generation program designed to delight children of all ages.

As editor of PC FIRING LINE/PC UNDERGROUND (alias Bill Salkin), I defy you to find the following topics discussed in any magazine—be it paper or disk: hardware maintenance, installing device drivers, critical-error handling, the EXEC function call, random file I/O, different memory models, bicubic splines, "printf" source code, and reducing the flab from IBM Pascal routines. High-sailing columns on Ada, Assembly, BASIC, C, FORTH, FORTRAN, LISP, Pascal, Hardware, and DOS! And NO paper magazine can supply FREEWARE, programmer-oriented DEMOS, and utilities in a ready-to-run format.



Bill Salkin

If you have been turned off by disk magazines, then look again. PCFL/PCUG—the magazine of the future—is available today for \$12 per single issue (half the price of our competitors' and we contain at least twice their content) or \$72 for a one-year (six-issue) subscription.

All funds in U.S. dollars. Foreign countries please send money orders and add \$5 (U.S.) airmail for each issue.

Requires an IBM-PC, 128K RAM, and a double-sided disk drive. We ship ONLY DS/DD disks.

**ABC Computing**  
 P.O. Box 5503, North Hollywood, CA 91616-5503 • (818) 509-9002

Guy M. Kelly

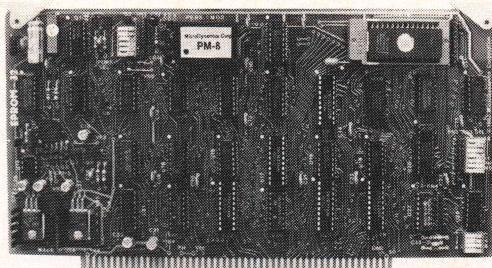
Gary M. Rader

Ken Holcombe

Microcompatibles

## S-100 EPROM PROGRAMMER

### EPROM-32



- Field-proven board meets IEEE-696 standard.
  - Programs 1K through 32K (byte) EPROMs.
  - Textool zero-insertion-force programming socket.
  - EPROM is programmed through I/O ports and can be verified through I/O ports or located in memory space for verification.
  - Programming voltage generated on-board.
  - Personality Modules adapt board to EPROMs:
- |           |      |           |       |            |
|-----------|------|-----------|-------|------------|
| PM-1-2508 | 2758 | PM-3-2732 | 2732A | PM-6-68764 |
| 2516      | 2716 | PM-4-2564 |       | PM-8-27128 |
| PM-2-2532 |      | PM-5-2764 |       | PM-9-27256 |
- Feature-packed CP/M-compatible control software includes fast programming algorithm.
- One year warranty.

\$269.95\*

(A & T)

**MicroDynamics**

Corporation

Suite 245 • 1355 Lynnfield Road • Memphis, TN 38119  
 (901)-682-4054

\* Price includes EPROM-32, documentation and two Personality Modules(specify). Additional Modules—\$7.95. Control software on 8" SSD diskette—\$29.95, UPS ground—\$2.00, UPS air—\$4.00. COD—\$1.65, foreign add \$15.00. VISA & MASTERCARD welcome.

See Dec. 1983 **Microsystems** for a review of the EPROM-32.

Circle no. 39 on reader service card.

Circle no. 1 on reader service card.

## Listing One

```

01C5 CD3604      CALL    ILPRT   ;PRINT MSG
01C8 46494C4520  DB      'FILE OPEN, READY TO RECEIVE',CR,LF,0

;
01E6 CD1802      RCVLP  CALL    RCVSECT ;GO RECEIVE A SECTOR
01E9 DAF801       JC      RCVEOT  ;CARRY SET-SECTOR IN OK
01EC CDBA03       CALL    WRSECT   ;WRITE TO DISK
01EF CDOC03       CALL    INCRSNO ;INCREMENT SECTOR +
01F2 CDCE02       CALL    SENDACK ;ALL OK - ACKNOWLEDGE
01F5 C3E601       JMP    RCVLP   ;GET NEXT SECTOR

;
01F8 CDBA03       RCVSECT CALL    WRSECT   ;EOF RCVD-OUTPUT ALL TO DISK
01FB CDCE02       CALL    SENDACK  ;SEND ACKNOWLEDGE
01FE CD9A03       CALL    CLOSFILE ;CLOSE DISK FILE
0201 CD4504       CALL    ERXIT    ;EXIT (NO ERROR)
0204 0D0A545241  DB      CR,LF,'TRANSFER COMPLETE$'

;
0218 AF          RCVSECT XRA    A      ;RECEIVE A SECTOR
0219 32EE04       STA    ERRCT   ;CLEAR ERROR CTR

;
021C 060A        RCVRPT MVI    B,10   ;LONG DELAY
021E CDE403       CALL    RECV    ;RECEIVE ONE CHARACTER
0221 DA3102       JC      RCVSERR ;NO CHAR - ERROR
0224 FE01         CPI    SOH    ;START OF HEADER?
0226 CA7802       JZ     RCVSOH  ;YES-CONTINUE RECEIVING
0229 B7          ORA    A      ;CLEAR CARRY FOR NEXT TRY

022A CA1C02       JZ     RCVRPT ;GO TRY AGAIN
022D FE04         CPI    EOT    ;END OF TRANSMISSION?
022F 37          STC    ;YES - CARRY=ALL DONE
0230 C8          RZ

;
0231 0601        RCVSERR MVI    B,1    ;REC ERROR-IGNORE REST OF TRANSMISSION
0233 CDE403       CALL    RECV    ;GET NEXT CHAR
0236 D23102       JNC    RCVSERR ;CHAR RECVD-GET ANOTHER
0239 3E15         MVI    A,NAK  ;NO CHAR-DATA ALL IN
023B CDD002       CALL    SEND    ;SEND NAK TO INDICATE REC ERROR
023E 3AEE04       LDA    ERRCT   ;GET ERROR CTR
0241 3C          INR    A      ;INCREMENT IT
0242 32EE04       STA    ERRCT   ;PUT IT BACK
0245 FE0A         CPI    ERRIM   ;TOO MANY?
0247 C21C02       JNZ    RCVRPT ;NO-TRY AGAIN

;
024A CD9A03       RCVSABT CALL    CLOSFILE
024D CD4504       CALL    ERXIT
0250 2B2B554E41  DB      '++UNABLE TO RECEIVE BLOCK'
0269 0D0A2B2B41  DB      CR,LF,'++ABORTING++$'

;
0278 0601        RCVSOH MVI    B,1    ;SOH RECEIVED
027A CDE403       CALL    RECV    ;SECTOR NUM IS NEXT
027D DA3102       JC      RCVSERR ;NO CHAR - ERROR
0280 57          MOV    D,A    ;SECTOR NUM TO D
0281 0601        MVI    B,1    ;SHORT WAIT
0283 CDE403       CALL    RECV    ;NEG OF SECTOR NUM IS NEXT
0286 DA3102       JC      RCVSERR ;NO CHAR - ERROR
0289 2F          CMA    ;MAKE POSITIVE
028A BA          CMP    D      ;SAME?
028B CA9102       JZ     RCVDATA ;YES - GET DATA
028E C33102       JMP    RCVSERR ;NOT SAME - ERROR

;
0291 7A          RCVDATA MOV    A,D    ;SECTOR NUM TO A
0292 32EC04       STA    RCVSNO ;SAVE FOR COMPARISON WITH EXPECTED
0295 0E00         MVI    C,O    ;CLEAR CHECKSUM
0297 218000       LXI    H,BASE+80H ;BUFFER ADDRESS

;
029A 0601        RCVCHR MVI    B,1    ;SHORT WAIT
029C CDE403       CALL    RECV    ;GET A CHARACTER

```

(Continued on page 72)

## CONVERT

If you want to move source files or documents from one type of disk to another ... you need CONVERT.

CONVERT operates on the IBM/PC and compatible computers. With it you can read, write and FORMAT over 40 types of CP/M diskettes.

CONVERT has been used to:

- o Move WordStar files from CP/M to PC/DOS diskettes.
- o Move Microsoft BASIC files from CP/M to PC/DOS diskettes.
- o "Publish" programs in many CP/M formats.

In independent reviews CONVERT was found to be superior to other conversion programs for quality, reliability and ease of use.

You can use CONVERT too. Available for \$99.00. VISA and MasterCard welcome. From:

**Selfware, Inc.**  
3545 Chain Bridge Rd.  
Suite 3  
Fairfax, VA 22030  
(703) 352-2977  
(800) 242-4355

Circle no. 62 on reader service card.

# Checks & Balances

A Professional Quality Z80/8080 Disassembler

## REVAS Version 3

Uses either ZILOG or 8080 mnemonics  
Includes UNDOCUMENTED Z80 opcodes  
Handles both BYTE (DB) & WORD (DW) data  
Disassembles object code up to 64k long!  
Lets you insert COMMENTS in the disassembly!

**A powerful command set gives you:**

INTERACTIVE disassembly  
Command Strings & Macros  
On-line Help  
Calculations in ANY Number Base!  
Flexible file and I/O control  
All the functions of REVAS V2.5

## REVAS:

Is fully supported with low cost user updates  
Runs in a Z80 CPU under CP/M\*  
Is normally supplied on SSSD 8" diskette  
Revax V 3...\$90.00 Manual only...\$15.00  
California Residents add 6½% sales tax

**REVASCO**  
**6032 Charlton Ave., Los Angeles, CA. 90056**  
**(213) 649-3575**

\*CP/M is a Trademark of Digital Research, Inc.

Circle no. 80 on reader service card.

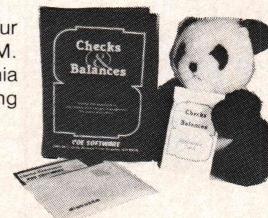
**HOW IMPORTANT IS  
YOUR CHECKBOOK?**  
Consider the alternatives when  
choosing a financial package

| CHECKS & BALANCES                   |  | DOLLARS & SENSE   | HOME ACCOUNTANT  |
|-------------------------------------|--|---|--|
| Price                               | \$74.95  | \$165.00  | \$150.00   |
| Going from one operation to another | Easily by entering a simple, easy to remember English command such as 'Enter'  | Must chain in and out of many menus in order to move between different operations | Like D&S, many levels of menus to traverse down, then back out and down another path                           |
| Correcting errors                   | All operations are screen oriented. Make corrections as with a word processor by typing over, inserting or deleting        | Not full screen. Only backs up one field at a time                                | Must stop work and chain to special correction menu and call up entry again, then through menus to resume work |
| Screen format for data entry        | Current and previous entry shown. Top screen scrolls to view account code as reminder. Data area looks like check register | One entry per screen. Must have on paper a list of accounts                       | Like D&S, one entry per screen which bears no resemblance to a checkbook or ledger                             |
| Space for payee and memo            | 39 characters for payee, 45 for memo with 39 extra for each extra disbursement memo  | 25 characters for payee. Program has no provision for memos                       | Only 15 characters each  |

Start the New Year out right (and get last year in order for tax time). Pick up **CHECKS & BALANCES** at your local dealer, or order direct from CDE. **CHECKS & BALANCES** is available for PC-DOS, MS-DOS and CP/M. Indicate computer type, operating system, and disk type (8" SSSD, 5½"SSDD, or 5½" DSSD). California residents include 6% sales tax, and everyone include \$2 shipping and handling. Dealer distribution bundling rates available. Write for complete catalogue of software for home and business.

Visa/MC accepted

**CDE SOFTWARE** (213)-661-2031  
**2463 McCready Ave. • Los Angeles, CA 90039**



**Listing One**

```

029F DA3102      JC     RCVSERR ;NO CHAR - ERROR
02A2 CD1204      CALL   TYPE   ;*****SECURITY BLANKET*****
02A5 77          MOV    M,A    ;CHAR TO BUFFER
02A6 2C          INR    L      ;POINT TO NEXT BUFFER SLOT
02A7 C29A02      JNZ    RCVCHR ;128 CHARS RECEIVED?
02AA 51          MOV    D,C    ;CHECKSUM TO D
02AB 0601      MVI    B,1    ;SHORT WAIT
02AD CDE403      CALL   RECV   ;GET CHECKSUM
02B0 DA3102      JC     RCVSERR ;NO CHAR - ERROR
02B3 BA          CMP    D      ;RECEIVED = COMPUTED?
02B4 C23102      JNZ    RCVSERR ;NOT SAME - ERROR
02B7 3AEC04      LDA    RCVSNO ;GET RECEIVED SECTOR NUM
02BA 47          MOV    B,A    ;INTO B FOR COMPARE
02BB 3AE004      LDA    SECTNO ;GET LAST SECTOR NUMBER
02BE B8          CMP    B      ;SAME?
02BF CAC802      JZ     RECVACK ;REPEAT - IGNORE
02C2 3C          INR    A      ;INCREMENT LAST
02C3 B8          CMP    B      ;SAME AS RECEIVED?

02C4 C2D402      JNZ    ABORT  ;NO - ERROR
02C7 C9          RET    ;RETURN TO MAIN LOOP
;
02C8 CDCE02      RECVACK CALL  SENDACK ;HANDSHAKE
02CB C31802      JMP    RCVSECT ;GET NEXT SECTOR
;
02CE 3E06      SENDACK MVI  A,ACK
;
02D0 CD9004      SEND   CALL   WRMDEM ;SEND CHAR TO MODEM
02D3 C9          RET
;
02D4 312B05      ABORT  LXI   SP,STCK ;RESTORE CP/M STACK POINTER
;
02D7 0601      ABORTL MVI  B,1    ;SHORT WAIT
02D9 CDE403      CALL   RECV   ;GET CHARACTER
02DC D2D702      JNC    ABORTL ;LOOP UNTIL INPUT STOPS
02DF 3E18          MVI    A,CAN  ;CANCEL CHARACTER
02E1 CDD002      CALL   SEND   ;SEND TO MODEM
;
02E4 0601      ABORTW MVI  B,1    ;SHORT WAIT
02E6 CDE403      CALL   RECV   ;GET CHARACTER
02E9 D2E402      JNC    ABORTW ;LOOP UNTIL INPUT STOPS
02EC 3E20          MVI    A,' ' ;GET A BLANK
02EE CDD002      CALL   SEND   ;SEND TO MODEM
02F1 CD4504      CALL   ERXIT  ;LEAVE (ERROR)
02F4 4D424F4F54        DB    'MBOOT PROGRAM CANCELLED$'
;
030C 3AE004      INCRSNO LDA  SECTNO ;GET CALCULATED SECTOR NUM
030F 3C          INR    A      ;INCREMENT IT
0310 32ED04      STA    SECTNO ;PUT IT BACK
0313 C9          RET
;
0314 115C00      ERASFIL LXI  D,FCB  ;FCB ADDRESS
0317 0E11          MVI    C,17   ;SEARCH FOR FIRST CODE
0319 CDO500      CALL   BDOS
031C 3C          INR    A      ;255 MEANS NO MATCH
031D C8          RZ    ;NOT FOUND IS OK
031E CD3604      CALL   ILPRT  ;PRINT MSG
0321 2B2B46494C        DB    'FILE EXISTS, TYPE Y TO ERASE:',0
0341 CD2A04      CALL   KEYIN  ;GET KEYBOARD CHAR
0344 F5          PUSH   PSW   ;SAVE A COPY
0345 CD1204      CALL   TYPE   ;ECHO IT
0348 CDB0B04      CALL   CRLF  ;CAR RET LINE FEED
034B F1          POP    PSW   ;GET CHAR BACK
034C E65F          ANI    5FH   ;REMOVE PARITY AND CASE
034E FE59          CPI    'Y'   ;Y = ERASE IT

```

(Continued on page 74)

# BACKUP PROTECTED SOFTWARE WITH COPY II MAC™

From the team who gave you COPY II PLUS and COPY II PC comes a new complete disk backup utility for your MACINTOSH computer. Features include:

- Bit Copy Program
- Make Files Visible/Invisible
- Sector/File Editor
- Format/Verify Disks
- Copy Protect/Unprotect
- Lock/Unlock Files
- Copy Files/Disk
- Rename File/Disk

Increase the power of your  
MACINTOSH . . . use **COPY II MAC**

Available at your local dealer or direct from us.

**CENTRAL POINT  
Software, Inc.**

ONLY  
**\$39.95**  
PLUS \$3.00 SHIPPING/HANDLING

9700 S.W. Capitol Highway, #100/Portland, OR 97219  
**(503) 244-5782**



WELCOME

(Prepayment Required)

This product is provided for the purpose of enabling you to make archival copies only.

Circle no. 6 on reader service card.



**NEW RELEASE**

## Eco-C Compiler Release 3.0

We think Rel. 3.0 of the Eco-C Compiler is the fastest full C available for the Z80 environment. Consider the evidence:

### Benchmarks\* (Seconds)

| Benchmark | <b>Eco-C</b> | Aztec | Q/C |
|-----------|--------------|-------|-----|
| Seive     | 29           | 33    | 40  |
| Fib       | 75           | 125   | 99  |
| Deref     | 19           | CNC   | 31  |
| Matmult   | 42           | 115   | N/A |

\*Times courtesy of Dr. David Clark  
CNC - Could Not Compile  
N/A - Does not support floating point

We've also expanded the library (120 functions), the user's manual and compile-time switches (including multiple non-fatal error messages). The price is still \$250.00 and includes Microsoft's MACRO 80. As an option, we will supply Eco-C with the SLR Systems assembler - linker - librarian for \$295.00 (up to six times faster than MACRO 80).

For additional information,  
call or write:



(317) 255-6476

6413 N. College Ave. • Indianapolis, Indiana 46220



## NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support.  
Full range transcendentals (tan, sin, cos, arctan, logs and exponentials)  
Data type conversion and I/O formatting.
- High level interrupt support.  
Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

**HS** / **FORTH**

• Fully Optimized & Tested for:  
IBM-PC IBM-XT IBM-JR  
COMPAQ EAGLE-PC-2  
TANDY 2000 CORONA  
LEADING EDGE

(Identical version runs on almost all MSDOS compatibles!)

- Graphics & Text (including windowed scrolling)
- Music - foreground and background includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler (interactive, easy to use & learn)
- Compare

BYTE Sieve Benchmark jan 83

HS/FORTH 47 sec BASIC 2000 sec

w/AUTO-OPT 9 sec Assembler 5 sec

other Forths (mostly 64k) 70-140 sec

FASTEAST FORTH SYSTEM

AVAILABLE.

TWICE AS FAST AS OTHER

FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$250.

Visa Mastercard  
Add \$10. shipping and handling

## HARVARD SOFTWORKS

P.O. Box 2579  
Springfield, OH 45501  
513/390-2087

Circle no. 35 on reader service card.

Circle no. 44 on reader service card.

**Listing One**

```

0350 C24B04      JNZ     MXIT    ;NOT Y - EXIT PROGRAM
0353 115C00      LXI     D,FCB   ;FCB ADDRESS
0356 0E13        MVI     C,19   ;DELETE FILE CODE
0358 CD0500      CALL    BDOS
;
035B 115C00      MAKEFIL LXI     D,FCB   ;FCB ADDRESS
035E 0E16        MVI     C,22   ;MAKE FILE CODE
0360 CD0500      CALL    BDOS
0363 3C          INR     A      ;A=255 IF ERROR
0364 C0          RNZ     ;RETURN - OK
0365 CD4504      CALL    ERXIT   ;LEAVE (ERROR)
0368 2B2B455252  DB     '++ERROR - CAN''T MAKE FILE',CR,LF

0383 4449524543  DB     'DIRECTORY MUST BE FULL$'
;
039A 115C00      CLOSFIL LXI     D,FCB   ;FCB ADDRESS
039D 0E10        MVI     C,16   ;FILE CLOSE CODE
039F CD0500      CALL    BDOS
03A2 3C          INR     A      ;A=255 IF ERROR
03A3 C0          RNZ     ;NOT ZERO IS OK
03A4 CD4504      CALL    ERXIT   ;LEAVE (ERROR)
03A7 2B2B43414E  DB     '++CAN''T CLOSE FILE$'
;
03BA =           WRSECT  EQU     $      ;WRITE SECTOR
03BA 115C00      LXI     D,FCB   ;FCB ADDRESS
03BD 0E15        MVI     C,21   ;WRITE SECTOR CODE
03BF CD0500      CALL    BDOS
03C2 B7          ORA     A      ;SET FLAGS
03C3 C2C703     JNZ     WRERR
03C6 C9          RET
;
03C7 CD3604      WRERR   CALL    ILPRT
03CA 2B2B455252  DB     '++ERROR WRITING FILE',CR,LF,0
03E1 C3D402     JMP     ABORT
;
; READ ONE CHARACTER FROM MODEM
;
03E4 D5          RECV    PUSH    D
03E5 CDB504      MWTI    CALL    RDMDDEM2 ;READ MODEM CHAR
03E8 D2F203     JNC     MCHAR  ;NO CARRY=CHAR RCVD
03EB 05          DCR     B      ;DECREMENT WAIT CTR
03EC C2E503     JNZ     MWTI   ;WAIT SOME MORE (IN 6510)
03EF D1          POP     D
03F0 37          STC     ;INDICATE ERROR
03F1 C9          RET
;
03F2 D1          MCHAR  POP     D
03F3 F5          PUSH    PSW    ;SAVE A COPY OF CHAR
03F4 81          ADD     C      ;ADD TO CHECKSUM
03F5 4F          MOV     C,A   ;PUT CKSUM BACK IN A
03F6 F1          POP     PSW    ;GET CHAR BACK
03F7 B7          ORA     A      ;CLEAR CARRY
03F8 C9          RET
;
03F9 2A0100      INITADR LHLD   BASE+1 ;GET XFER BASE ADDRESS
03FC 110300      LXI     D,3    ;GET OFFSET
03FF 19          DAD     D      ;ADD OFFSET FOR STAT
0400 222304      SHLD   VSTAT+1 ;PUT IN CALL
0403 19          DAD     D      ;ADD OFFSET TO KEYIN
0404 222E04      SHLD   VKEYIN+1
0407 19          DAD     D      ;ADD OFFSET TO TYPE
0408 221804      SHLD   VTYPET+1
;
```

(Continued on page 76)

UNIX

**STEP UP TO ELEGANCE**  
 \*UNIX is the environment professionals  
 depend on for productivity,  
 portability, flexibility  
 and simply making  
 programming a  
 pleasure.

LET YOUR MODEM CONNECT YOU TO...

# THE SOLUTION™

- EXPANSIVE SOFTWARE DEVELOPMENT FACILITIES. Language and Operating System design.
- LANGUAGES INCLUDE: C, Fortran 77, RATFOR, COBOL, SNOBOL, BS, Assembler + LISP.
- USENET Bulletin Board System—800+ networked international UNIX sites, 190+ categories, 300+ new articles per day.
- Inter/intra system mail + communications.
- UNIFY: for professional data-base application development.
- UNIX & System enhancements from U.C. Berkeley and Korsmeyer Electronic Design Inc.
- Online UNIX manuals + Expert consultation available.
- Just a local modem call from 300+ cities nationwide via Telenet.
- Complete photo typesetting service.
- FAST and LOW COST, low as \$8.95 hr. connect.
- \$24.95 = 1 hr. FREE system time, SOLUTION News subscription, BYTE BOOK: Introducing The UNIX System, 556 pp.

\*UNIX is a trademark of Bell Labs.  
**korsmeyer**  
 ELECTRONIC DESIGN, INC.



Payment via VISA or  
MasterCard

5701 Prescott Avenue Lincoln, NE 68506-5155  
402/483-2238 10a-7p Central

Circle no. 54 on reader service card.



## MULTI-TASKING for the IBM AT! also for the IBM PC or XT with MULTI-JOB™

"MULTI-JOB the most cost effective choice for the user with a need for multi-tasking." PC Age Vol. 3.8

With MULTI-JOB software up to 9 IBM PC DOS compatible programs can be running at the same time. Example, have your communication program running in the background, and still be using your word processing, spreadsheet programs, etc., at the same time! The keyboard and screen can be assigned to any job with a simple keystroke. The remaining jobs will continue to run unattended. With the many different options, MULTI-JOB is a very powerful package.

- \* No special hardware is required.
- \* Allows priorities to be given between each job.
- \* Programs can be run simultaneously or one at a time.
- \* 30-day free trial period.

|                           |          |
|---------------------------|----------|
| <b>MULTI-JOB</b>          | \$159.00 |
| <b>ELECTRONIC DISK</b>    | \$ 49.00 |
| <b>SPOOL PROGRAM</b>      | \$ 24.00 |
| <b>SET MEMORY UTILITY</b> | \$ 24.00 |

B&L COMPUTER CONSULTANTS, 7337 Northview,  
Suite B, Boise, ID 83704, (208) 377-8088.



Dealer's inquiries are welcome. Call or write for a free catalog.

Circle no. 11 on reader service card.

# LISP FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE  
OF ARTIFICIAL  
INTELLIGENCE FOR  
YOUR IBM PC.

## ■ DATA TYPES

Lists and Symbols  
Unlimited Precision Integers  
Floating Point Numbers  
Character Strings  
Multidimensional Arrays  
Files  
Machine Language Code

## ■ MEMORY MANAGEMENT

Full Memory Space Supported  
Dynamic Allocation  
Compacting Garbage Collector

## ■ FUNCTION TYPES

EXPR/FEXPR/MACRO  
Machine Language Primitives  
Over 190 Primitive Functions

## ■ IO SUPPORT

Multiple Display Windows  
Cursor Control  
All Function Keys Supported  
Read and Splice Macros  
Disk Files

## ■ POWERFUL ERROR RECOVERY

## ■ 8087 SUPPORT

## ■ COLOR GRAPHICS

## ■ LISP LIBRARY

Structured Programming Macros  
Editor and Formatter  
Package Support  
Debugging Functions  
.OBJ File Loader

## ■ RUNS UNDER PC-DOS 1.1 or 2.0

## IQLISP

5 1/4" Diskette \_\_\_\_\_ \$175.00  
and Manual \_\_\_\_\_  
Manual Only \_\_\_\_\_ \$ 30.00

*∫ q Integral Quality*

P.O. Box 31970  
Seattle, Washington 98103-0070  
(206) 527-2918

Washington State residents add sales tax.  
VISA and MASTERCARD accepted.  
Shipping included for prepaid orders.

## Listing One

```

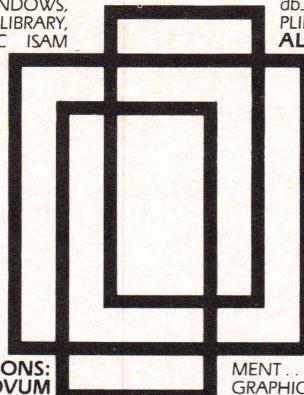
;          ;
040B 3E0D  CRLF  MVI    A,CR   ;CARRAIGE RET
040D CD1204  CALL   TYPE
0410 3E0A    MVI    A,LF   ;LINE FEED
;
;          ;
0412 F5      TYPE   PUSH   PSW   ;TYPE A CHARACTER
0413 C5      PUSH   B
0414 D5      PUSH   D
;
0415 E5      PUSH   H
0416 4F      MOV    C,A
0417 CD0000  VTYPY  CALL   $-$   ;CALL BDOS
041A E1      POP    H
041B D1      POP    D
041C C1      POP    B
041D F1      POP    PSW
041E C9      RET
;
;          ;
041F C5      STAT   PUSH   B   ;CONSOLE STATUS
0420 D5      PUSH   D
0421 E5      PUSH   H
0422 CD0000  VSTAT  CALL   $-$   ;CALL BDOS
0425 E1      POP    H
0426 D1      POP    D
0427 C1      POP    B
0428 B7      ORA    A
0429 C9      RET
;
;          ;
042A C5      KEYIN  PUSH   B   ;READ A KEY
042B D5      PUSH   D
042C E5      PUSH   H
042D CD0000  VKEYIN CALL   $-$   ;CALL BDOS
0430 E1      POP    H
0431 D1      POP    D
0432 C1      POP    B
0433 E67F    ANI    7FH
0435 C9      RET
;
;          ;
0436 E3      ILPRT  XTHL   ;TYPE A LINE
;
;          ;
0437 7E      ILPLP  MOV    A,M   ;GET CHARACTER
0438 B7      ORA    A   ;SET FLAGS
0439 CA4304  JZ     ILPRET ;ZERO MEANS DONE
043C CD1204  CALL   TYPE   ;TYPE CHARACTER
043F 23      INX    H   ;NEXT MEM LOC
0440 C33704  JMP    ILPLP ;NEXT CHARACTER
;
;          ;
0443 E3      ILPRET XTHL   ;MODIFIED RETURN TO STACK
0444 C9      RET
;
;          ;
0445 D1      ERXIT  POP    D   ;EXIT (NOT ALWAYS ERROR)
0446 0E09    MVI    C,9   ;PRINT EXIT MESSAGE
0448 CD0500  CALL   BDOS
;
;          ;
044B CD3604  MXIT   CALL   ILPRT
044E 0D0A444F4E DB    CR,LF,'DON''T FORGET TO DISCONNECT MODEM'
0470 0D0A00  DB    CR,LF,O
;
;          ;
0473 CD8704  EXIT   CALL   CLMDEM ;CLOSE RS-232 FILE
0476 CD0B04  CALL   CRLF  ;PRINT A FINAL CRLF
0479 2A2B05  LHLD   STCK  ;CP/M STACK POINTER
047C F9      SPHL
047D C9      RET   ;BACK INTO SP
;
;          ;
;          ; 6510 SETUP ROUTINES
;
;          ;
047E 3E05  OPMDEM MVI    A,5   ;FILE OPEN CMD

```

(Continued on page 78)

# Once you choose Lattice, our friends will C you through...

**LATTICE INC.:** LATTICE WINDOWS, CURSES UNIX SCREEN CONTROL LIBRARY, C-FOOD SMORGASBORD, dB-C ISAM COMPATIBLE WITH dBASE II AND III... **LIFEBOAT ASSOCIATES:** FLOAT 87 8087 SUPPORT PACKAGE, HALO GRAPHICS PACKAGE, PANEL SCREEN LIBRARY... **GREENLEAF SOFTWARE:** THE GREENLEAF C FUNCTIONS... **C SOURCE:** BASICLC C FUNCTIONS FOR BASIC USER... **SOFTCRAFT:** BTREIVE ISAM FILE SYSTEM, BTREIVE ISAM NETWORK FILE SYSTEM... **BLAISE COMPUTING:** TOOLS, TOOLS2, VIEW MANAGER SCREEN PACKAGE... **MORNING STAR SYSTEMS:** PROLIBRARY, PROSCREEN... **CREATIVE SOLUTIONS:** WINDOWS FOR C... **NOVUM ORGANUM:** C POWERS PACKS, MATHEMATICS POWER PACKS, ADVANCED POWER PACKS, DATABASE POWER PACKS, TELECOMMUNICATIONS POWER PACKS W/ SOURCE... **PHACT ASSOCIATES:** PHACT ISAM LIBRARY... **RAIMA CORPORATION:**



db... VISTA DBMS... **PHOENIX:** PLINK86, PFILE86... **RELATIONAL DATABASE SYSTEMS:** C-ISAM FILE ACCESS METHOD... **MINDBANK:** V-FILE VIRTUAL MEMORY/FILE SYSTEM... **HUNTER & READY:** VRTX C INTERFACE LIBRARY... **GRAPHIC SOFTWARE SYSTEMS:** GSS DRIVERS, GSS TOOLKIT KERNEL SYSTEM... **OPT-TECH DATA PROCESSING:** OPT-TECH SORT... **ACCUDATA SOFTWARE:** C-TREE ISAM, C-SORT SORT... **TRIO SYSTEMS:** C-INDEX+ ISAM... **COMPU CRAFT:** C VIEW FORMS/WINDOW MANAGEMENT... **SCIENTIFIC ENDEAVORS:** GRAPHIC PRESENTATION SCIENTIFIC GRAPHICS... **LEMMA SYSTEMS, ESSENTIAL SOFTWARE,** INC.: C LIBRARY... **SOFTWARE LABS:** C UTILITIES PACKAGE... **FAIRCOM:** C-tree BY FAIRCOM ISAM WITH SOURCE

Contact Lattice to learn how we can help your C program development.



**LATTICE®**

P.O. Box 3072  
Glen Ellyn, IL 60138  
312/858-7950  
TWX 910-291-2190

Circle no. 58 on reader service card.

# NEW!

**RELOCATABLE  
Z-80 MACRO  
ASSEMBLER  
FROM MITEK**

### It's a real bargain! Here's why:

- Only \$49.95 plus shipping
- 8080 to Z-80 Source Code Converter
- Generates Microsoft compatible REL files or INTEL compatible hex files
- Compatible with Digital Research macro assemblers MAC & RMAC
- Generates Digital Research compatible SYM files
- Full Zilog mnemonics
- INCLUDE and MACLIB files
- Conditional assembly
- Separate data, program, common and absolute program spaces
- Customize the Macro Assembler to your requirements with installation program
- Over 3 times faster than Microsoft M80 macro assembler
- Z-80 Linker and Library Manager for Microsoft compatible REL files available as a total package with Macro Assembler for only \$95.00
- Manual only is \$15

**TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804**  
For information or technical assistance: 1-808-623-6361

Specify desired 5 1/4" or 8" soft-sector format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed. Include \$5 for postage and handling.

**MITEK** P.O. Box 2151  
Honolulu, HI 96805

Z-80 is a trademark of Zilog, Inc. MAC, RMAC, and ZSID are trademarks of Digital Research, Inc. M80 is a trademark of Microsoft Corp.

Circle no. 66 on reader service card.

# Pascal and C Programmers

Your programs can now compile the **FirsTime™**

**FirsTime** is an intelligent editor that knows the rules of the language being programmed. It checks your statements as you enter them, and if it spots a mistake, it identifies it. **FirsTime** then positions the cursor over the error so you can correct it easily. **FirsTime** will identify all syntax errors, undefined variables, and even statements with mismatched variable types. In fact, any program developed with the **FirsTime** editor will compile on the first try.

### More than a syntax checker!

**FirsTime** has many unique features found in no other editor. These powerful capabilities include a zoom command that allows you to examine the structure of your program, automatic program formatting, and block transforms.

If you wish, you can work even faster by automatically generating program structures with a single key-stroke. This feature is especially useful to those learning a new language, or to those who often switch between different languages.

**Other Features:** Full screen editing, horizontal scrolling, function key menus, help screens, inserts, deletes, appends, searches, and global replacing.

Programmers enjoy using **FirsTime**. It allows them to concentrate on program logic without having to worry about coding details. Debugging is reduced dramatically, and deadlines are more easily met.

|                            |       |
|----------------------------|-------|
| <b>FirsTime</b> for PASCAL | \$245 |
| <b>FirsTime</b> for C      | \$295 |
| Microsoft PASCAL Compiler  | \$245 |
| Microsoft C Compiler       | \$395 |
| Demonstration disk         | \$25  |

Get an extra **\$100 off** the compiler when it is purchased with **FirsTime**. (N.J. residents please add 6% sales tax.)

**Spruce**  
**Technology Corporation**

110 Whispering Pines Drive  
Lincroft, N.J. 07738

(201) 741-8188 or (201) 663-0063

Dealer enquiries welcome. Custom versions for computer manufacturers and language developers are available.

**FirsTime** is a trademark of Spruce Technology Corporation.



Circle no. 65 on reader service card.

## Listing One

|             |         |      |        |                            |
|-------------|---------|------|--------|----------------------------|
| 0480 320306 |         | STA  | MDFUNC | ;STORE FOR 6510            |
| 0483 CDD804 |         | CALL | G06510 | ;TURN ON 6510              |
| 0486 C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 0487 3E04   | CLMDEM  | MVI  | A,4    | ;CLOSE FILE CODE           |
| 0489 320306 |         | STA  | MDFUNC | ;STORE FOR 6510            |
| 048C CDD804 |         | CALL | G06510 | ;TURN ON 6510              |
| 048F C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 0490 F5     | WRMDEM  | PUSH | PSW    | ;SAVE A COPY OF CHAR       |
| 0491 320406 |         | STA  | MDCHAR | ;SAVE FOR 6510             |
| 0494 3E03   |         | MVI  | A,3    | ;WRITE MODEM CODE          |
| 0496 320306 |         | STA  | MDFUNC | ;STORE FOR 6510            |
| 0499 CDD804 |         | CALL | G06510 | ;START 6510                |
| 049C F1     |         | POP  | PSW    | ;GET CHAR BACK             |
| 049D C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 049E 3E01   | RDMDEM  | MVI  | A,1    | ;READ FUNCTION             |
| 04A0 320306 |         | STA  | MDFUNC | ;STORE FOR 6510            |
| 04A3 CDD804 |         | CALL | G06510 | ;START 6510                |
| 04A6 3A0506 |         | LDA  | MDREC  | ;SEE IF CHAR RCVD          |
| 04A9 FE00   |         | CPI  | 0      | ;0 - NONE RECEIVED         |
| 04AB CAB304 |         | JZ   | RDMNC  | ;GO SET CARRY              |
| 04AE 3A0406 |         | LDA  | MDCHAR | ;GET CHARACTER             |
| 04B1 B7     |         | ORA  | A      | ;SHOW CHAR RCVD            |
| 04B2 C9     |         | RET  |        |                            |
| 04B3 37     | RDMNC   | STC  |        | ;SET - NO CHAR             |
| 04B4 C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 04B5 3E02   | RDMDEM2 | MVI  | A,2    | ;READ WITH WAIT FUNCTION   |
| 04B7 320306 |         | STA  | MDFUNC | ;STORE FOR 6510            |
| 04BA CDD804 |         | CALL | G06510 | ;START 6510                |
| 04BD 3A0506 |         | LDA  | MDREC  | ;SEE IF CHAR RCVD          |
| 04C0 FE00   |         | CPI  | 0      | ;0 - NONE RECEIVED         |
| 04C2 CACA04 |         | JZ   | RDMNC2 | ;GO SET CARRY              |
| 04C5 3A0406 |         | LDA  | MDCHAR | ;GET MODEM CHAR            |
| 04C8 B7     |         | ORA  | A      | ;SHOW CHAR RCVD            |
| 04C9 C9     |         | RET  |        |                            |
| 04CA 37     | RDMNC2  | STC  |        | ;CARRY - NO CHAR           |
| 04CB C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 04CC 3E06   | KYBD    | MVI  | A,6    | ;READ KEYBOARD FUNCTION    |
| 04CE 320306 |         | STA  | MDFUNC | ;STORE FOR 6510            |
| 04D1 CDD804 |         | CALL | G06510 | ;TURN ON 6510              |
| 04D4 3A0406 |         | LDA  | MDCHAR | ;GET CHARACTER             |
| 04D7 C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 04D8 E5     | G06510  | PUSH | H      | ;SAVE HL                   |
| 04D9 210016 |         | LXI  | H,MD65 | ;GET 6510 RTNE ADDR        |
| 04DC 2206F9 |         | SHLD | BIOSAD | ;STORE FOR BIOS65          |
| 04DF 3E09   |         | MVI  | A,9    | ;BIOS CODE TO CALL SUBPROG |
| 04E1 3200F9 |         | STA  | BIOSFN | ;STORE FOR 6510            |
| 04E4 3E01   |         | MVI  | A,1    | ;1 TURNS ON 6510           |
| 04E6 3200CE |         | STA  | ON6510 | ;TURN IT ON                |
| 04E9 00     |         | NOP  |        | ;REQD FOR HARDWARE         |
| 04EA E1     |         | POP  | H      | ;RESTORE HL                |
| 04EB C9     |         | RET  |        |                            |
| ;           |         |      |        |                            |
| 04EC 00     | RCVSNO  | DB   | 0      |                            |
| 04ED 00     | SECTNO  | DB   | 0      |                            |
| 04EE 00     | ERRCT   | DB   | 0      |                            |
| 04EF        |         | DS   | 60     |                            |
| 052B        | STCK    | DS   | 2      |                            |
| ;           |         |      |        |                            |
| 052D        |         | END  |        |                            |

End Listing One

## *Listing Two*

CPMMD65.TX

| LINE# | LOC  | CODE                             | LINE                            |
|-------|------|----------------------------------|---------------------------------|
| 00001 | 0000 |                                  | ; ****                          |
| 00002 | 0000 |                                  | ;                               |
| 00003 | 0000 |                                  | ; MODEM OR RS-232 HANDLER       |
| 00004 | 0000 |                                  | ; FOR USE WITH CP/M MBOOT       |
| 00005 | 0000 |                                  | ;                               |
| 00006 | 0000 |                                  | ; W.G. PIOTROWSKI               |
| 00007 | 0000 |                                  | ;                               |
| 00008 | 0000 |                                  | ; ****                          |
| 00009 | 0000 |                                  | ;                               |
| 00010 | 0000 |                                  | ; EQUATES                       |
| 00011 | 0000 |                                  | ;                               |
| 00012 | 0000 | FILE =128                        | ; FILE NUMBER                   |
| 00013 | 0000 | ;                                |                                 |
| 00014 | 0000 | SCNKEY =\$FF9F                   | ; KERNAL ROUTINES               |
| 00015 | 0000 | SETLFS =\$FFBA                   | ;                               |
| 00016 | 0000 | SETNAM =\$FFBD                   | ;                               |
| 00017 | 0000 | OPEN =\$FFC0                     | ;                               |
| 00018 | 0000 | CLOSE =\$FFC3                    | ;                               |
| 00019 | 0000 | CHKIN =\$FFC6                    | ;                               |
| 00020 | 0000 | CHKOUT =\$FFC9                   | ;                               |
| 00021 | 0000 | CLRCHN =\$FFCC                   | ;                               |
| 00022 | 0000 | CHROUT =\$FFD2                   | ;                               |
| 00023 | 0000 | GETIN =\$FFE4                    | ;                               |
| 00024 | 0000 | ;                                |                                 |
| 00025 | 0000 | RIBUF =\$F7                      | ; RS-232 INPUT BUF ADD          |
| 00026 | 0000 | ROBUF =\$F9                      | ; RS-232 OUTPUT BUF ADD         |
| 00027 | 0000 | RIDBS =\$29B                     | ; RS-232 BUF END PTR            |
| 00028 | 0000 | RIDBE =\$29C                     | ; RS-232 BUF STRT PTR           |
| 00029 | 0000 | ENABL =\$2A1                     | ; RS-232 ACTIVE                 |
| 00030 | 0000 | ;                                |                                 |
| 00031 | 0000 | XLATE =\$2100                    | ; C64 TO ASCII TABLE            |
| 00032 | 0000 | ·BUFIN =\$2200                   | ; RS-232 INPUT BUFFER           |
| 00033 | 0000 | BUFOUT =\$2300                   | ; RS-232 OUTPUT BUFFER          |
| 00034 | 0000 | ;                                |                                 |
| 00035 | 0000 | *=\$1600                         |                                 |
| 00036 | 1600 | ;                                |                                 |
| 00037 | 1600 | 4C 06 16 MODEM                   | JMP START                       |
| 00038 | 1603 | 00 FUNC                          | .BYTE 0 ; FUNCTION CODE         |
| 00039 | 1604 | 00 CHAR                          | .BYTE 0 ; CHARACTER IN/OUT      |
| 00040 | 1605 | 00 CHREC                         | .BYTE 0 ; RS-232 CHAR RCVD FLAG |
| 00041 | 1606 | ;                                |                                 |
| 00042 | 1606 | AD 03 16 START                   | LDA FUNC ; GET FUNCTION         |
| 00043 | 1609 | 18 CLC                           | ; CLEAR FOR ADD                 |
| 00044 | 160A | 6D 03 16 ADC FUNC                | ; *2 FOR TABLE ACCESS           |
| 00045 | 160D | AA TAX                           |                                 |
| 00046 | 160E | BD 6B 17 LDA ADRTBL-2,X          | ; GET LOW ADDRESS               |
| 00047 | 1611 | 8D 1B 16 STA JMPSUB+1            | ; PUT IN JMP                    |
| 00048 | 1614 | BD 6C 17 LDA ADRTBL-1,X          | ; GET HI ADDRESS                |
| 00049 | 1617 | 8D 1C 16 STA JMPSUB+2            | ; PUT IN JMP                    |
| 00050 | 161A | 4C 00 00 JMPSUB JMP 0            | ; JMP TO SUBPROGRAM             |
| 00052 | 161D | ;                                |                                 |
| 00053 | 161D | ; CLOSE RS-232 FILE AND CLEAN UP |                                 |
| 00054 | 161D | ;                                |                                 |
| 00055 | 161D | A9 80 CLOSIT LDA #FILE           | ; FILE NUMBER                   |
| 00056 | 161F | 20 C3 FF JSR CLOSE               | ; KERNAL                        |
| 00057 | 1622 | 20 57 16 JSR CLRKBF              | ; CLEAR KEYBOARD BUFFER         |
| 00058 | 1625 | 60 RTS                           |                                 |
| 00059 | 1626 | ;                                |                                 |
| 00060 | 1626 | ; OPEN RS-232 FILE AND DO SETUP  |                                 |
| 00061 | 1626 | ;                                |                                 |
| 00062 | 1626 | 20 1D 16 OPENIT JSR CLOSIT       | ; CLOSE-IN CASE                 |
| 00063 | 1629 | A9 80 LDA #FILE                  | ; FILE NUMBER                   |
| 00064 | 162B | A2 02 LDY #2                     | ; RS-232 DEVICE                 |
| 00065 | 162D | A0 FF LDY #\$FF                  | ; NO COMMAND                    |
| 00066 | 162F | 20 BA FF JSR SETLFS              | ; CALL KERNAL                   |

(Continued on next page)

## Listing Two

```

00067 1632 A9 02          LDA #2           ;TWO CHAR NAME
00068 1634 A2 7A          LDX #<SET232    ;LO ADDRESS
00069 1636 A0 17          LDY #>SET232    ;HI ADDRESS
00070 1638 20 BD FF       JSR SETNAM      ;KERNEL
00071 163B 20 C0 FF       JSR OPEN        ;KERNEL
00072 163E A2 00          LDX #<BUFIN     ;RS232 INBUF LO ADDR
00073 1640 A0 22          LDY #>BUFIN     ; HI ADDR
00074 1642 86 F7          STX RIBUF      ;MOVE IT
00075 1644 84 F8          STY RIBUF+1
00076 1646 A2 00          LDX #<BUFOUT    ;RS232 OUTBUF LO ADDR
00077 1648 A0 23          LDY #>BUFOUT    ; HI ADDR
00078 164A 86 F9          STX ROBUF      ;MOVE IT
00079 164C 84 FA          STY ROBUF+1
00080 164E 20 0E 17       JSR BLDXTB     ;BUILD XLATE TABLE
00081 1651 A9 00          LDA #0
00082 1653 8D 7E 17       STA LSTCHR     ;SHOW LAST CHAR AS A NULL
00083 1656 60              RTS
00084 1657
00085 1657               ; EMPTY KEYBOARD BUFFER
00086 1657
00087 1657 20 9F FF       CLRKBF JSR SCNKEY   ;KEY PRESSED?
00088 165A 20 E4 FF       JSR GETIN      ;GET A CHARACTER
00089 165D C9 00          CMP #0         ;IS IT A NULL
00090 165F D0 F6          BNE CLRKBF    ;NOT A NULL - GET NEXT
00091 1661 60              RTS
00092 1662
00093 1662               ; CHARACTER OUTPUT TO RS-232
00094 1662
00095 1662 A2 80          OUTPUT LDX #FILE   ;GET FILE NUM
00096 1664 20 C9 FF       JSR CHROUT    ;OPEN FOR OUTPUT
00097 1667 90 06          BCC OTOK      ;CARRY SET IS ERROR
00098 1669 20 26 16       JSR OPENIT    ;MUST BE CLOSED
00099 166C 4C 62 16       JMP OUTPUT    ;TRY AGAIN
00100 166F AD 04 16       OTOK LDA CHAR   ;GET CHARACTER AGAIN
00101 1672 20 D2 FF       JSR CHRROUT   ;KERNEL
00102 1675 AD A1 02       WAITOT LDA ENABL  ;GET STATUS
00103 1678 29 01          AND #1        ;STILL RUNNING BIT
00104 167A D0 F9          BNE WAITOT   ;HANG UNTIL DONE
00105 167C 20 CC FF       JSR CLRCHN   ;CLEAR CHANNEL
00106 167F 60              RTS
00107 1680
00108 1680               ; CHARACTER INPUT FROM RS-232
00109 1680
00110 1680 A2 80          INPUT  LDX #FILE   ;FILE NUMBER
00111 1682 20 C6 FF       JSR CHKIN    ;OPEN FOR INPUT
00112 1685 90 06          BCC INOK      ;CARRY SET IS ERROR
00113 1687 20 26 16       JSR OPENIT   ;MUST BE CLOSED
00114 168A 4C 80 16       JMP INPUT    ;TRY AGAIN
00115 168D A9 00          INOK LDA #0     ;ZERO FOR CLEAR
00116 168F 8D 05 16       STA CHREC   ;SHOW NO CHAR RCVD
00117 1692 AD A1 02       LDA ENABL   ;STATUS BYTE
00118 1695 29 02          AND #2        ;RECEIVING BIT
00119 1697 D0 10          BNE WAIT    ;RUNNING - WAIT
00120 1699 AD 9B 02       LDA RIDBS   ;GET BUF START PTR
00121 169C CD 9C 02       CMP RIDBE   ;COMPARE TO END
00122 169F F0 1D          BEQ INEX    ;NOTHING IN BUFFER
00123 16A1 A9 01          LDA #1        ;ONE FOR SET
00124 16A3 8D 05 16       STA CHREC   ;SHOW CHAR RECEIVED
00125 16A6 4C B8 16       JMP INGET   ;GET THE CHAR
00126 16A9 AD A1 02       WAIT LDA ENABL  ;STATUS BYTE
00127 16AC 29 02          AND #2        ;RECEIVING BIT
00128 16AE F0 08          BEQ INGET   ;NOT RUNNING - EXIT
00129 16B0 A9 01          LDA #1        ;ONE FOR SET
00130 16B2 8D 05 16       STA CHREC   ;SHOW CHAR RECEIVED
00131 16B5 4C A9 16       JMP WAIT    ;WAIT UNTIL IN
00132 16B8 20 E4 FF       INGET JSR GETIN  ;GET CHARACTER
00133 16BB 8D 04 16       STA CHAR    ;STORE FOR CP/M

```

(Continued on page 82)

# "Ouvrez les fenêtres!"\*

Introducing **MATIS**, the powerful new developmental system from France.

A complete and meticulously detailed program to make a programmer's work easier, faster, and... but of course... better.

- Window Management Systems
- Screen Generator
- Expanded Basic Commands
- Can be accessed from other languages
- 100% Assembler
- Automatic Scrolling in Windows
- Virtual Page larger than screen (up to 65534 rows x 65534 columns)
- Save or Print Pages
- MS-DOS
- 170 Page Manual (In English Mon Ami!)
- Only \$150.

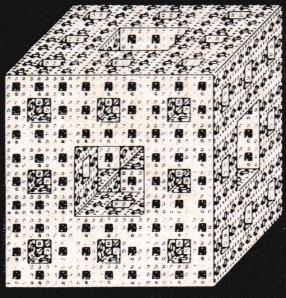
ORDER BY MAIL—WRITE OR CALL FOR COMPLETE DESCRIPTION  
No license fee.

## Softway, Inc.

500 Sutter Street • Suite 222-M • San Francisco, CA 94102  
Tel: (415) 397-4666 Telex: 880857

### \* "Open the windows!"

Circle no. 92 on reader service card.



## WALTZ LISP<sup>(TM)</sup>

The one and only **adult** Lisp system for CP/M users.

Waltz Lisp is a very powerful and complete implementation of the Lisp programming language. It includes features previously available only in large Lisp systems. In fact, Waltz is substantially compatible with Franz (the Lisp running under Unix), and is similar to MacLisp. Waltz is perfect for Artificial Intelligence programming. It is also most suitable for general applications.

**Much** faster than other microcomputer Lisps. • Long integers (up to 611 digits). Selectable radix • True dynamic character strings. Full string operations including fast matching/extraction. • Flexibly implemented random file access. • Binary files. • Standard CP/M devices. • Access to disk directories. • Functions of type lambda (expr), nlambda (fexpr), leexpr, macro. • Splicing and non-splicing character macros. • User control over all aspects of the interpreter. • Built-in prettyprinting and formatting facilities. • Complete set of error handling and debugging functions including user programmable processing of undefined function references. • Virtual function definitions. • Optional automatic loading of initialization file. • Powerful CP/M command line parsing. • Fast sorting/merging using user defined comparison predicates. • Full suite of mapping functions, iterators, etc. • Assembly language interface. • Over 250 functions in total. • The best documentation ever produced for a micro Lisp (300+ full size pages, hundreds of illustrative examples).

Waltz Lisp requires CP/M 2.2, Z80 and 48K RAM (more recommended). All common 5" and 8" disk formats available.

**PC**  
**PRO CODE**  
(TM)  
**INTERNATIONAL**

15930 SW Colony Pl.  
Portland, OR 97224

Unix® Bell Laboratories.  
CP/M® Digital Research Corp.

**Version 4.4**

(Now includes Tiny Prolog  
written in Waltz Lisp.)

**\$169\***

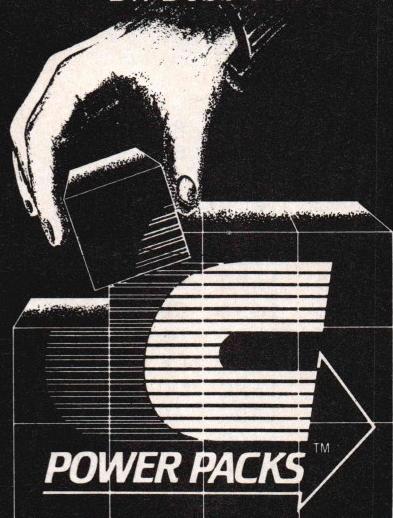
\*Manual only: \$30 (refundable with order). All foreign orders: add \$5 for surface mail, \$20 for airmail. COD add \$3. Apple CP/M and hard sector formats add \$15.

Call free **1-800-LIP-4000** Dept. #11  
In Oregon and outside USA call 1-503-684-3000

Circle no. 73 on reader service card.

"This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal



## COMPLETE SOURCES

- PACK 1: Building Blocks I** \$149  
250 Functions: DOS, Printer, Video, Asynch
- PACK 2: Database** \$399  
100 Functions: B-Trees, Variable Records
- PACK 3: Communications** \$149  
135 Functions: Smart-modem™, Xon/Xoff, Modem-7, X-Modem
- PACK 4: Building Blocks II** \$149  
100 Functions: Dates, Text Windows, Pull-down Menus Data Compression
- PACK 5: Mathematics I** \$99  
35 Functions: Log, Trig, Square Root
- PACK 6: Utilities I** \$99  
Archive, Diff, Replace, Scan, Wipe (Executable Files only)

Lattice™, Microsoft™, DeSmet™,  
CI-86™ Compilers on IBM PC/XT/AT™  
Small and Large Memory Models.  
Credit cards accepted  
(\$7.00 handling/Mass. add 5%)

**SH** SOFTWARE HORIZONS INC.

165 Bedford Street  
Burlington, Mass. 01803  
(617) 273-4711

NOVUM ORGANUM

Circle no. 90 on reader service card.

## Listing Two

```

00134 16BE 20 CC FF    INEX   JSR CLRCHN
00135 16C1 60          RTS
00136 16C2
;
;   CHARACTER INPUT FROM RS-232 WITH WAIT LOOP
00137 16C2
;
;   INPUT2 LDA #20      ;OUTER LOOP LIMIT
00139 16C2 A9 14      INPUT2 STA OTCTR ;PUT IN LOOP CONTROL LOC
00140 16C4 8D 7D 17    LDA #0   ;INNER LOOP COUNTS AROUND
00141 16C7 A9 00      STA INCTR ;PUT IN LOOP CONTROL LOC
00142 16C9 8D 7C 17    IN2LP  JSR INPUT ;SEE IF THERE'S A CHARACTER
00143 16CC 20 80 16    LDA CHREC ;GET RECEIVED CHAR
00144 16CF AD 05 16    BNE IN2EX ;0 - NONE RECEIVED
00145 16D2 D0 0A      DEC INCTR ;COUNT DOWN INNER LOOP
00146 16D4 CE 7C 17    BNE IN2LP ;WAIT
00147 16D7 D0 F3      DEC OTCTR ;COUNT DOWN OUTER LOOP
00148 16D9 CE 7D 17    BNE IN2LP ;WAIT SOME MORE
00149 16DC D0 EE      IN2EX  RTS
00150 16DE 60
;
;   INPUT KEYBOARD CHARACTER AND DEBOUNCE
00151 16DF
;
;   KBDCHR JSR SCNKEY ;SCAN THE KEYBOARD
00154 16DF 20 9F FF    JSR GETIN ;GET THE CHARACTER
00155 16E2 20 E4 FF    CMP LSTCHR ;SAME AS LAST CHAR?
00156 16E5 CD 7E 17    BNE KBNCHR ;NO - NEW CHARACTER
00157 16E8 D0 05      LDA #0   ;SAME - NEED A NULL
00158 16EA A9 00      JMP KBDBNC ;GO TO DEBOUNCE LOOP
00159 16EC 4C F2 16    KBNCHR STA LSTCHR ;NEW CHAR - SAVE IT
00160 16EF 8D 7E 17    KBDBNC LDY #7  ;DEBOUNCE OUTER LOOP
00161 16F2 A0 07
;
;   LDY #0      ;INNER LOOP INIT
00162 16F4 A2 00      CMP #0   ;REPEATING CHAR LO LIM
00163 16F6 C9 00      BEQ KBDDLY ;NULL IS OK
00164 16F8 F0 06      CMP #33  ;SPACE IS HI LIM
00165 16FA C9 21      BCS KBDDLY ;33 OR MORE IS OK
00166 16FC B0 02      LDY #50  ;LONG DELAY FOR RPTG CHAR
00167 16FE A0 32      KBDDLY DEX ;DELAY LOOP
00168 1700 CA
00169 1701 D0 FD      BNE KBDDLY ;INNER LOOP
00170 1703 88          DEY
00171 1704 D0 FA      BNE KBDDLY ;OUTER LOOP
00172 1706 AA          TAX ;CHARACTER TO A
00173 1707 BD 00 21    LDA XLATE,X ;GET ASCII VALUE
00174 170A 8D 04 16    STA CHAR ;STORE FOR Z-80
00175 170D 60          RTS
00176 170E
;
;   BUILD ASCII TRANSLATION TABLE
00177 170E
;
;   BLDXTB LDY #0      ;LOOP TO CLEAR TABLE
00179 170E A2 00      LDX #0
00180 1710 A9 00      LDA #0
00181 1712 9D 00 21    BLDX1 STA XLATE,X
00182 1715 E8          INX
00183 1716 D0 FA      BNE BLDX1 ;LOOP
00184 1718
;
;   LDY #0      ;START INDEX
00185 1718 A2 00      LDX #0
00186 171A A0 41      LDY #65 ;STOP INDEX+1
00187 171C A9 00      LDA #0
00188 171E 20 5D 17    JSR MVINTB ;NULL
00189 1721 A2 41      LDX #65 ;LOWER CASE
00190 1723 A0 5B      LDY #91
00191 1725 A9 61      LDA #97
00192 1727 20 5D 17    JSR MVINTB
00193 172A A2 5B      LDX #91 ;MISC CHARACTERS
00194 172C A0 60      LDY #96
00195 172E A9 5B      LDA #91
00196 1730 20 5D 17    JSR MVINTB
00197 1733 A2 C1      LDX #193 ;UPPER CASE
00198 1735 A0 DB      LDY #219
00199 1737 A9 41      LDA #65

```

(Continued on page 84)

## QUALITY SOFTWARE AT REASONABLE PRICES

CP/M Software by  
**Poor Person Software**

**Poor Person's Spooler** \$49.95  
All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.

**Poor Person's Spread Sheet** \$29.95  
Flexible screen formats and BASIC-like language. Pre-programmed applications include Real Estate Evaluation.

**Poor Person's Spelling Checker** \$29.95  
Simple and fast! 33,000 word dictionary. Checks any CP/M text file.

**aMAZEing Game** \$29.95  
Arcade action for CP/M! Evade goblins and collect treasure.

**Crossword Game** \$39.95  
Teach spelling and build vocabulary. Fun and challenging.

**Mailing Label Printer** \$29.95  
Select and print labels in many formats.

**Window System** \$29.95  
Application control of independent virtual screens.

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add \$5 handling charge. California residents include sales tax.

**Poor Person Software**  
3721 Starr King Circle  
Palo Alto, CA 94306  
tel 415-493-3735

CP/M is a registered trademark of Digital Research

Circle no. 71 on reader service card.

## ASSEMBLE 3-6 TIMES FASTER ON THE IBM PC

Introducing **FAST ASSEM-86™**, the first Editor/Assembler for the IBM PC and PC compatibles. **FAST ASSEM-86™ (FASM)** is significantly faster and easier to use than the IBM Macro-Assembler (MASM). Whether you are new to assembly language and want to quickly write a small assembly language routine, or are an experienced MASM user tired of waiting months to assemble large files, **FAST ASSEM-86** will bring the excitement back to assembly language.

### FAST ASSEM-86 IS MUCH FASTER:

- How fast is **FASM™**? The graph below shows relative assembly times for a 48K source file. For large files like this we blow MASM's doors off at 3 times their speed. For smaller 8K files we positively vaporize them at 6 times their speed.

**FASM™** (110 sec.)  
MASM v1.0 (340 sec.)

- **FAST ASSEM-86** is faster for the following reasons: (1) Written entirely in assembly language (unlike MASM). (2) Editor, assembler and source file always in memory so you can go instantly from editing to assembling and back. (3) Eliminates the time needed to LINK programs. Executable .COM files can be created directly. (Also creates .OBJ files completely compatible with the IBM linker).

### FAST ASSEM-86 IS EASIER TO USE:

**FASM** includes many other features to make your programming simpler.

- Listings are sent directly to screen or printer. Assemblies can be single stepped and examined without having to leave the editor.
- Access the built in cross reference utility from the editor.
- Full support of 186 and 286 (real mode) instructions.
- Both Microsoft and 8087 floating point formats are supported. 8087 and 287 instructions supported directly without macros for faster assembly.
- Calculator mode: Do math in any radix even using symbols from the symbol table.
- Direct to memory assembly feature lets you test execute your code from editor.
- Coming soon: A coordinated symbolic debugger.

**COMPATIBILITY:** **FASM** is designed for source code compatibility with MASM and supports most of its important features.

**Introductory Price \$199**

**Speedware™**

IBM, Microsoft trademarks of IBM Corp., Microsoft Corp. respectively.

Dealer inquiries welcome  
916-966-6247

Box D1, 2931 Northrop Avenue  
Sacramento, CA 95825

Circle no. 97 on reader service card.

## You Read Dr.Dobb's Journal And You Don't Subscribe?!

Save over \$23.00 off newsstand prices for 2 yrs.

Save over \$10.00 for 1 yr.

Can you afford to miss an issue with information vital to your interests? As a subscriber you can look forward to articles on Small-C, FORTH, CP/M, S-100, Compiler optimization, Concurrent Programming and more, delivered right to your door. And you'll never miss the issue that covers your project.

**Yes!** Sign me up for

2 yrs. \$47       1 yr. \$25

I enclose a check/money order

Charge my Visa, MasterCard,

American Express

Please bill me later

Name \_\_\_\_\_  
Address \_\_\_\_\_

Credit Card \_\_\_\_\_ Exp. date \_\_\_\_\_ Zip \_\_\_\_\_

Account No. \_\_\_\_\_

Signature \_\_\_\_\_

## Listing Two

```

00200 1739 20 5D 17      JSR MVINTB
00201 173C A9 00          LDA #0
00202 173E 8D 1C 21      STA XLATE+28
00203 1741 8D 1D 21      STA XLATE+29
00204 1744 8D 1E 21      STA XLATE+30
00205 1747 8D 1F 21      STA XLATE+31
00206 174A A9 08          LDA #8      ;BACKSPACE
00207 174C 8D 14 21      STA XLATE+20 ;INST DEL
00208 174F 8D 94 21      STA XLATE+148 ;INST DEL UC
00209 1752 A9 0D          LDA #13    ;CR
00210 1754 8D 8D 21      STA XLATE+141 ;RETURN UC
00211 1757 A9 20          LDA #32    ;SPACE
00212 1759 8D A0 21      STA XLATE+160 ;SPACE UC
00213 175C 60             RTS

00214 175D               ;
00215 175D 8C 79 17      MVINTB STY MXSTOP ;SAVE STOP VAL IN MEMORY
00216 1760 9D 00 21      MVINT1 STA XLATE,X ;PUT ASCII EQUIV IN TABLE
00217 1763 18             CLC      ;CLEAR FOR ADD
00218 1764 69 01          ADC #1    ;NEXT CHAR CODE
00219 1766 E8             INX      ;NEXT TABLE SLOT
00220 1767 EC 79 17      CPX MXSTOP ;COMPARE TO STOP VALUE
00221 176A D0 F4          BNE MVINT1 ;LOOP
00222 176C 60             RTS

00223 176D               ;
00224 176D               ; DATA
00225 176D               ;
00226 176D 80 16          ADRTBL .WORD INPUT ;SUBPROG XFER TABLE
00227 176F C2 16          .WORD INPUT2 ; IN ORDER OF REQ CODE
00228 1771 62 16          .WORD OUTPUT
00229 1773 1D 16          .WORD CLOSIT
00230 1775 26 16          .WORD OPENIT
00231 1777 DF 16          .WORD KBDCHR
00232 1779 00             MXSTOP .BYTE 0
00233 177A 06             SET232 .BYTE 6,0 ;RS232 PARAMS
00233 177B 00
00234 177C 00             INCTR  .BYTE 0 ;LOOP CTR - INPUT2
00235 177D 00             OTCTR  .BYTE 0 ;LOOP CTR - INPUT2
00236 177E 00             LSTCHR .BYTE 0 ;LAST KEYBOARD CHAR
00237 177F               .END

```

ERRORS = 00000

## SYMBOL TABLE

| SYMBOL  | VALUE |
|---------|-------|
| ADRTBL  | 176D  |
| BUFOUT  | 2300  |
| CHREC   | 1605  |
| CLRCHN  | FFCC  |
| FUNC    | 1603  |
| INCTR   | 177C  |
| INPUT   | 1680  |
| KBDCHR  | 16DF  |
| MODEM   | 1600  |
| OPEN    | FFC0  |
| OUTPUT  | 1662  |
| ROBUF   | 00F9  |
| SETNAM  | FFBD  |
| XLATE   | 2100  |
| BLDX1   | 1712  |
| CHAR    | 1604  |
| CHROUT  | FFD2  |
| CLRKBF  | 1657  |
| GETIN   | FFE4  |
| INEX    | 16BE  |
| INPUT2  | 16C2  |
| KBDLBY  | 1700  |
| MVINT1  | 1760  |
| OPENIT  | 1626  |
| RIBUF   | 00F7  |
| SCNKEY  | FF9F  |
| START   | 1406  |
| BLDXTB  | 170E  |
| CHKIN   | FFC6  |
| CLOSE   | FFC3  |
| ENABL   | 02A1  |
| IN2EX   | 16DE  |
| INGET   | 16B8  |
| JMPSUB  | 161A  |
| KBNCHR  | 16EF  |
| MVINTB  | 175D  |
| OTCTR   | 177D  |
| RIDBE   | 029C  |
| SET232  | 177A  |
| WAIT    | 16A9  |
| BUFIN   | 2200  |
| CHKOUT  | FFC9  |
| CLOSIT  | 161D  |
| FILE    | 0080  |
| IN2LP   | 16CC  |
| INOK    | 168D  |
| KBDDBNC | 16F2  |
| LSTCHR  | 177E  |
| MXSTOP  | 1779  |
| OTOK    | 166F  |
| RIDBS   | 029B  |
| SETLFS  | FFBA  |
| WAITOT  | 1675  |

END OF ASSEMBLY

End Listing

# ADVERTISERS!

## Don't Miss March... Dr. Dobb's Journal Special Artificial Intelligence Issue

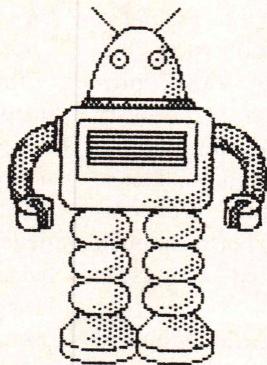
Space Reservation Deadline: January 4, 1985

Materials Deadline: January 11, 1985

### CONTACT

Walter Andrzejewski  
Shawn Horst  
(415) 424-0600

Beth Dudas  
(714) 643-9439



Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Circle no. 72 on reader service card.



## MacTech

Vol. 1 No. 2

The Macintosh Programming Journal

Nov 84

### AT LAST!

A no-nonsense, no fluff journal devoted to programming **FOR** the Mac, **ON** the Mac!

Let MacTech's editorial board teach you the Macintosh technology of windows, quickdraw, events and resources. We have assembled a team of professionals to uncover and explain Mac's secrets.

### ASSEMBLY LAB

by David Smith

### C WORKSHOP

by Bob Denny

### PASCAL

by Chris Derossi

### MACBASIC

by Dave Kelly

### MACFORTH

by Joerg Langowski



5 years of  
experience in  
PC Computer  
Education!  
Your symbol  
of excellence!

YES! I want to program my Macintosh! Send me a one year subscription. I enclose \$24 US or \$36 Overseas.

MacTech, P.O. Box 846, Placentia, CA 92670  
(714) 993-9939

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Circle no. 91 on reader service card.

## Scroll & Recall™

Screen and Keyboard Enhancement  
for the IBM - PC, XT and Compatibles

Allows you to conveniently scroll back through data that has gone off the top of your display screen.

Allows you to easily recall and edit your previously entered DOS commands and data lines.

Very easy to use, fully documented. Compatible with all versions of DOS, monochrome & graphic displays.

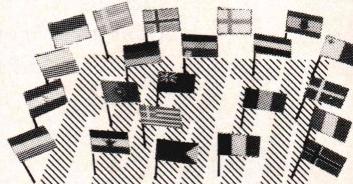
\$69 - Visa, M/C, Check, COD, POs  
Phone orders accepted

### Make Your Work Easier!

To Order or to Receive Additional Information, Write or Call:

**Opt-Tech Data Processing**  
P.O. Box 2167 • Humble, Texas 77347  
(713) 454-7428  
Dealer Inquiries Welcome

Circle no. 68 on reader service card.



## FORTH into Europe

Support for major FORTHS and our own products

### VAX-FORTH 32

- ★ Complete VMS support
- ★ Command line qualifiers
- ★ DEC compatible full screen editor
- ★ On line HELP facilities
- ★ Start-up files
- ★ Switchable log-files
- ★ System files with precompiled modules
- ★ Cross compilers available for most microprocessors

### FORTH-83 CROSS-COMPILERS

- ★ B-tree symbol table of unlimited size
- ★ Compiles FORTH-83 nucleus
- ★ Compiles 16 or 32 bit code
- ★ Two passes allow automatic pruning of nucleus for ROM applications
- ★ Automatic handling of defining words
- ★ Targets include 1802, Z8, 8070, 8080, 6801/3, 6502, 6511Q, 6809, 99xxx, 8086/8, 68000, Z80

MicroProcessor Engineering, 21 Hanley Road, Shirley,  
Southampton, SO1 5AP, England, Tel: 0703 780084

Die FORTH-Systeme Angelika Flesch, Schuetzenstrasse 3,  
7820 Titisee-Neustadt, West Germany, Tel: 07651 1665

Circle no. 45 on reader service card.

# Unstructured Forth Programming

## An Introduction

by Richard Wilton

One of the features of Forth is that it is a "structured" language. This means that Forth programs are written in variously sized chunks that fit together like a Chinese box puzzle. This is in contrast to "unstructured" Fortran or BASIC programs, which are written in variously sized chunks that fit together like the noodles on a plate of spaghetti. This is what makes "structured" programming languages such as Forth so much superior to "unstructured" languages such as Fortran.

Nowadays, Fortran has acquired roughly the same status among recent computer science graduates as Middle English or Aramaic. Nevertheless, there are still some of us around who not only remember what Fortran is, but have actually written programs in it. In fact, there are still a few people who think that "Forth" is just an abbreviation for IBM's Fortran 4 (H) compiler.

Let's face it: despite the universal

structures, were somehow built into Forth? For that matter, wouldn't it be nice every once in a while to be able to fall back on that old Fortran standby, the computed GOTO?

Of course it would! It is the purpose of this little article to fill in the gaps in the pristine structure of Forth, to repair the glaring omissions we've just mentioned, and to restore a little unstructured sanity to the deeply nested, block-structured world of the dedicated Forth hacker. Note: all examples are written in Forth-83. FIG-Forth and Forth-79 users will have to adjust the "tick" and ROLL references accordingly.

### The GOTO Statement

```
: GOTO ( cfa --- ) R> DROP  
EXECUTE ;
```

This little gem of a definition expects the code field address of a Forth definition on the stack. You'll note, however,

**"What Forth programmer doesn't long for a simple, unstructured, unconditional GOTO every once in awhile?"**

scorn heaped upon it by "modern" structured programming disciples, good old Fortran has its strong points. In fact, in certain situations, the obligatory structured nature of Forth makes elegant programming impossible. What Forth programmer doesn't long for a simple, unstructured, unconditional GOTO every once in a while? Who doesn't wish that the nice, straightforward arithmetic IF, which can be expressed so concisely in Fortran or in two or three machine in-

structions, were somehow built into Forth? For that matter, wouldn't it be nice every once in a while to be able to fall back on that old Fortran standby, the computed GOTO?

### The Arithmetic IF Statement

```
: AIF ( cfa_A cfa_B cfa_C n --- )  
? DUP 0=  
IF ROT  
ELSE 0>  
IF ROT ROT  
THEN
```

Richard Wilton, Laboratory Microsystems, P.O. Box 10430, Marina Del Rey, CA 90295.

THEN 2DROP GOTO ;

The arithmetic IF statement is obvious in concept: transfer control to point A if n is negative, to point B if it's zero, and to point C if it's positive. This is the sort of thing you do all the time in assembly language:

```
OR AX,AX  
JS POINT_A  
JZ POINT_B  
POINT_C:
```

But just try to express that succinctly in Forth:

```
DUP 0< IF ['] POINT_A  
ELSE 0 = IF ['] POINT_B  
ELSE ['] POINT_C  
THEN  
THEN  
EXECUTE
```

Pretty clumsy, right? Now here's the elegant solution, which is concise and straightforward because it ignores the dogma of block-structuring:

```
: EXAMPLE ( n --- )  
    ['] POINT_A
```

```
['] POINT_B  
['] POINT_C  
3 ROLL AIF ;
```

### The Computed GOTO Statement

As a final example, we implement Fortran's familiar

```
GOTO (x1,x2,x3,...,xn),i
```

a control statement so elegant that it was copied almost verbatim by the creators of BASIC. Making it work in Forth requires setting up a table of code field addresses in advance. Then you simply index the table and GOTO the right address.

```
: CGOTO ( cfa_table n --- ) 2* +  
    @ GOTO ;
```

The utility of using CGOTO, rather than a viper's nest of IF-ELSE-THEN's or CASE statements, is obvious to any competent programmer and is not worth belaboring here.

The more astute reader might observe at this point, "But what about statement numbers?" Well, folks, if

you think about it for a moment, you'll realize that statement numbers make good sense in Forth. For one thing, numbered statements would make it easy to do "source-directed" editing, thus eliminating the need for those clumsy, space-wasting screen files. Also, silly philosophical debates about the "proper" names for Forth definitions would be unnecessary if Forth definitions were numbered instead of named.

By now it should be obvious how much can be gained by writing unstructured Forth programs. No doubt, further improvements could be obtained by incorporating elements of PL/I or even COBOL into Forth.

Of course, such improvements must be left as an exercise for the reader. We're too busy adapting the syntax of Forth to fit onto 80-column punched cards.

DDJ

#### Reader Ballot

Vote for your favorite feature/article.  
Circle Reader Service No. 195.

*Elegance  
Power  
Speed*



**C Users' Group**  
Supporting All C Users  
Box 97  
415 Euclid  
McPherson, Kansas 67460  
(316) 241-1065

Circle no. 17 on reader service card.

## The Little Board™...\$349\*

The world's simplest and least expensive CP/M computer



**CP/M 2.2 INCLUDED**

\*UNDER \$200 IN OEM QUANTITIES

- 4 MHz Z80A CPU, 64K RAM, Z80A CTC, 2732 Boot ROM
- Mini/Micro Floppy controller (1-4 Drives, Single/Double Density, 1-2 sided, 40/80 track)
- Only 5.75 x 7.75 inches, mounts directly to a 5 1/4" floppy drive
- 2 RS232C Serial Ports (75-9600 baud & 75-38,400 baud), 1 Centronics Printer Port
- Power Requirement: +5VDC at .75A; +12VDC at .05A/On-board -12V converter
- CP/M 2.2 BDOS • ZCPR3 CCP • Enhanced AMPRO BIOS
- AMPRO Utilities included:
  - read/write to more than 2 dozen other formats (Kaypro, Televideo, IBM CP/M86....)
  - format disks for more than a dozen other computers
  - menu-based system customization
- BIOS and Utilities Source Code Available
- SCSI/PLUS Adapter:
  - Mounts directly to Little Board • Slave I/O board control • Full ANSC X3T9.2
  - 16 bidirectional I/O lines • \$99/Quantity 1

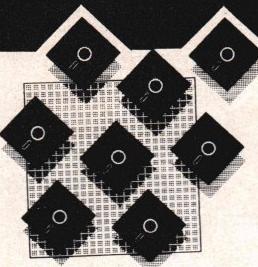
Distributor/Dealer/Reps  
Inquiries Invited

Z80A is a registered trademark of Zilog, Inc.  
CP/M is a registered trademark of Digital Research.

**AMPRO**  
COMPUTERS. INCORPORATED

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

Circle no. 7 on reader service card.



## **APL\*PLUS/PC, Version 3.1; TOOLS, Volume 1; FINAN- CIAL & STATISTICAL LIBRARY, Version 1.0**

**Company:** STSC, Inc., 2115 East Jefferson St., Rockville, MD 20852

**Computer:** 192K IBM PC (or compatible) with one or more disk drives; **TOOLS** requires 256K and MSDOS Version 2.0 or later

**Price:** \$595 for APL\*PLUS/PC; \$295 for TOOLS; \$275 for LIBRARY

**Circle Reader Service No. 129**

**Reviewed by Stefan H. Unger**

I spent several months wondering which advanced language I would get for my PC. At the time there was a lot of press about C, but in the back of my mind I really wanted to get APL. The best APL was reputedly from STSC, Inc., but I felt it was a little too expensive for my home use.

I had programmed in APL since about 1972, mostly statistical routines that I needed in my work in computer-aided drug design. In fact, I had even published some APL functions, but I was certainly not what you would call a hard-core convert to APL. This language simply offered me—as a chemist and not as a programmer—the greatest freedom to be creative without worrying about the stultifying detail required by most other languages. At work I originally had used a STSC APL time-sharing service, switched to another APL time-sharing service for several years, then ported everything to our inhouse IBM 3081 VS APL 4.0 system. However, recently I have been busy with computer graphics and quantum/molecular mechanics and not statistics (i.e., APL) at all.

Having been exposed to the ease and elegance of APL, I found that learning

BASIC for my PC was more self-flagellation than enlightenment. It was slow, awkward, and, well, silly (Isn't it silly to have to dimension variables? After all, computers should do that kind of stuff for you....).

Turbo Pascal—with its easier user interface—was not yet getting much press, and the language didn't seem to offer any particular advantages over Fortran or BASIC in general approach. So I bought a C compiler, knowing C might come in handy at work. At least it would run faster. Unfortunately, it was a disaster. I couldn't accept the amount of time it took to compile and link after each minor debugging step, and, worst of all, I couldn't really make much sense of the I/O function calls... it all seemed so needlessly complex.

At this low point in my experience with C, I was given the opportunity of reviewing STSC's APL\*PLUS/PC system for *DDJ*. The irony of this is that *DDJ* is such a staunch advocate of C (which means that I have now enraged most of my audience). Of course, I'm not proposing that APL and C are equivalent or even on the same level. For example, some implementations of APL have been done in C (e.g., Dyalog APL), but not vice versa. (A recent article by D. Saunders, "Unix, C and APL," *Unix/World*, 1[6], 59, 1984, in fact argues that the two languages are naturally complementary.) But for the average PC applications developer, is APL worth a serious look? Read on!

First of all, I think learning well-defined APL symbols is less ridiculous than trying to match braces in C or remembering the dozens of function calls with their nonstandard syntax and location. The only real problem with APL is the character set and some of the difficulties that it causes in graphics/printing interfaces. Of course, C "gets closer to the machine," but STSC APL\*PLUS/PC allows interfaces

to outside assembly code, DOS, DOS files, interrupts, peeking/poking, and so on. It is also much easier to learn, debug, and use, being what Saunders calls a "very high-level language." From descriptions of the user interface to Turbo Pascal (full-screen editing, pointers between source and object code, fast compilations), I would say that APL is at least as good as—if not considerably better than—Turbo Pascal in this regard.

### **The System**

The APL\*PLUS/PC system is supplied on two diskettes, and a special APL character ROM replaces the one on your graphics card (the IBM version of APL uses software-generated characters and requires both a color monitor and 8087). Special ROMs are available for IBM, AT&T, Compaq, Columbia, Eagle, Hyperion and Televideo microcomputers. Corona is supported, but without graphics, Seequa is supported without communications and Wang is supported without graphics and sound. The IBM AT, other MSDOS computers and the DEC Rainbow are scheduled for support. (Although all of these machines are "supported", there might be some tradeoffs in performance. Not all graphics boards are supported, either; check carefully.)

Only the least useful characters of the original character set have been replaced by the special APL characters, so the system has virtually no impact on ordinary word processing, spreadsheets, and so on. The system requires 192K and PC or MSDOS with one or more disk drives. Documentation is four IBM-style loose-leaf binders with sections on installation, user's guide, introductory tutorial, formatting, files, programmer's manual, system functions, and a comprehensive index. Color-coded APL keyboard labels are supplied—the tackiest part of the

package. The Gilman and Rose textbook classic *APL: An Interactive Approach* and a quarterly newsletter are also provided. You also get on the STSC mailing list, which will definitely keep your mailbox full.

Other sources of information are available for additional cost: two volumes of *Collected Whizbangs* by Roy Sykes, Jr. (a fantastic collection of advanced tricks) and a volume entitled *APL in Practice: What You Need to Know to Install and Use Successful APL Systems and Major Applications*, edited by Rose and Schick (to assuage any remaining doubt that APL is not taken seriously by the business world). The Rose and Schick book is directed toward mainframe APL uses, but the overview is good and a number of chapters contain some generally useful advice, particularly in the third part of the book: "The Core of APL."

STSC has also announced the first collection of application development TOOLS (\$295, requires 256K, PC or MSDOS at 2.0 or later and APL\* PLUS/PC at 3.0 or later), which provides communications, screen management, report and output formatting, disk and file management, software development tools, and two games.

Additionally, STSC has developed a Financial and Statistical Library (\$275) for the PC, containing many primitive functions that can be incorporated into application programs; this means no window dressing on the output, although "dressed" versions are also available for some of the functions. The package is overpriced for what you get. For example, my favorite correlation coefficient program is faster than the one supplied in STATISTI; not all of the terms are defined in the manual (e.g., Durbin-Watson Statistic); there doesn't appear to be any ANOVA, a common feature of most statistical packages (at least not in the index nor any of the obvious places); and we get the program FED-TAX79, my favorite . . . it is 1984, isn't it? The list goes on.

A new addition to STSC's product line is Pocket APL, which is a simplified and more affordable version of APL (only \$95). It requires 128K RAM but uses a soft character set (instead of the special ROM) for color systems and the keyword APL system for

monochrome. Communications, graphics, full files and workspaces, and the full-screen editor are not supported, nor is there an interface to DOS. However, it does supply on-line calculator mode, full-screen cursor control, on-line help, a file system, □ FMT formatting, error trapping features, and over 50 system functions. STSC apparently believes that Pocket APL will be the Turbo of the APL world.

STSC has also solved the major problem for applications developers who want to use APL, namely the requirement of a full APL interpreter. Because the cost of the STSC system is at least \$595, this used to be a major

drawback. Now, a Run Time System is available for a license fee of from \$50 to \$100, depending upon quantity and possibly your negotiating skills. The Run Time System allows applications packages that buffer the user from APL; the user has no APL directly available. Neither calculator mode nor definition-editing-display of functions is available. In fact, the APL character ROM is not required, and the keyboard stays in text mode. The full-screen editor can be used only to edit variables. Applications developers are offered a software publishing program with royalties of "5 to 15 percent."

Generally, all of the documentation

## CP/M-80 C Programmers . . .

# Save time

... with the BDS C Compiler. Compile, link and execute faster than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast*, then it's

time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

### BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/Z80 CP/M command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for runtime segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.
- Plus . . .
- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner and the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CP/M system utilities, games and more.
- A nationwide BDS C User's Group (\$10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C is heralded for its remarkable speed.

BYTE Magazine placed BDS C ahead of all other 8080/Z80 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost twice as

fast as its closest competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."

Tim Pugh, Jr.  
in *InfoWorld*  
"Performance: Excellent.  
Documentation: Excellent.  
Ease of Use: Excellent."

*InfoWorld*  
Software Report Card  
"...a superior buy..."  
Van Court Hare  
in *Lifelines/The Software Magazine*

*Don't waste another minute on a slow language processor. Order your BDS C Compiler today!*

Complete Package (two 8" SSDD disks, 181-page manual): \$150  
Free shipping on prepaid orders inside USA.  
VISA/MC, COD's, rush orders accepted.  
Call for information on other disk formats.

BDS C is designed for use with CP/M-80 operating systems, version 2.2 or higher. It is not currently available for CP/M-86 or MS-DOS.

BD Software, Inc.  
P.O. Box 2368  
Cambridge, MA 02238  
(617) 576-3828

# BD Software

Circle no. 12 on reader service card.

is very well done with good quality paper, typesetting, editing, and so on. Of course, nothing is perfect. The Gilman and Rose textbook unfortunately is weakest on the most difficult aspects of APL (a consistent theme, as you shall see), and there are a number of minor editing problems. The front cover proclaims "Includes APL2 and APL for PC's," but this is not really explained—in a forward, for example. In fact, I didn't know what APL2 was and eagerly awaited the revelation inside. Alas, the best I could do was to figure out that it was an IBM product. The index didn't help either. There were only a few references to "APL for PC's" inside the text, despite the 1983 publication date, and a reference to "see inside the back cover for details" in the preface . . . very funny! It's blank.

Two serious deficiencies in the package are the lack of quick reference cards and the need for more detailed tutorial material on graphics, memory, and machine language features—the most difficult aspects of the extended language. This makes it tough on the neophyte or convert. A workspace demo on graphics was so confusing that I couldn't make it do anything useful. Documentation for the interrupt handler  $\square INT$  gives only a few examples, a stern warning about "improperly constructed arguments," and a reference to the section on  $\square CALL$ . Documentation for the latter system function is more extensive but still difficult to follow.

To whet the microsystem user's appetite, I should mention that STSC has announced two interesting developments: the maxi/mini APL\*PLUS system will be available under Unix, which means speed, nested arrays, and a good multiuser environment, and an APL "compiler" for IBM mainframe VM/CMS and MVS/TSO will be available, promising a three-fold enhancement in speed. (There are quotes around compiler because you still must run the compiled functions under the standard APL environment.) STSC is the most active, broadest-based APL vendor in the world, which I hope speaks well for its commitment toward APL for the PC.

Finally, even as I write this review of Version 3.1, Version 4.0 is being announced. Included is documentation on

all workspaces, new stickers, and a placard with the APL character set. Speedups for  $\square LOAD$ ,  $\square COPY$ ,  $\square DEF$ , and  $\square SS$  (substring search), exponentiation, membership, and inner product are implemented (some up to 10 times faster). A soft character set for use on an IBM color monitor with IBM color graphics card eliminates the special APL ROM—at the considerable sacrifice of off-screen scrolling and full-screen editor. Several new APL keyboard facilities, new enhancements to the full-screen editor, graphics, and some new language features (e.g., enhanced validity checking of certain primitives) are implemented. The workspaces have all been "reviewed and revised." Communications using the smart terminal mode have been improved to allow, for example, the host to execute a line in the PC. An upgrade for existing users will be about \$90.

### Features

STSC APL\*PLUS/PC is a nearly complete version of standard APL, lacking only some of the newer extensions such as nested arrays and some of the extensions of operators found in the IBM mainframe APL2 or STSC's Unix-based system for maxis/minis. I will review a large number of the existing enhancements.

A keyword mode is available through a function key toggle; in this mode, APL symbols are replaced by English words. An on-line HELP facility can be customized for use in applications. User-definable function keys are available through a system function.

Workspace size expands to available memory. On my 256K system a  $\square CLEAR$  workspace is 112,336 bytes, but memory can be partitioned for a non-APL functionality under DOS; for example, TOOLS supplies a RAM disk program (although I find the AST Superdrive much easier to use). Maximum object size is 65,536 bytes, including overhead. In a clear workspace, this translates to about 8,190 floating-point and 32,761 integer numbers. Floating-point numbers are eight bytes (17 significant digits in range  $+/- 1.797 \dots E308$ ); both integer and Boolean are two bytes (range  $+/- 32767$ ). No bit packers need apply!

An optional 8087 math coprocessor is automatically detected by the system

and can be toggled on/off with a  $\square POKE$ . It significantly speeds up exponential, transcendental, and other functions. See the section on benchmarks.

High-resolution black-and-white or color graphics and sound are supported with system functions. Use of the graphics primitives is nontrivial, dangerous in certain circumstances (misuse can blitz your graphics board!), and lacks a good tutorial description. The only documentation is in the system function outline and a brief section in the user's guide. Using the session parameter  $E=64$  to reduce "snow" will cause a significant slowdown in the speed of color graphics; without this setting, color graphics literally flash on the screen, but with the setting, it takes about 3 seconds to paint a full screen. The sound primitive  $\square SOUND$  is extremely easy to use, and a fascinating music-generating program in one of the workspaces produces pleasing little ditties. Simple windows and screen scrolling control are available; the TOOLS I package offers further enhancements such as menu processing, maintenance, screen input facility, and forms maintenance.

Communications facilities for uploading and downloading from mainframe or other computers comprise several system functions, plus a number of programs in one workspace as illustrative examples. Additional 3278 and IRMA facilities are in TOOLS I. One of STSC's ads shows an 11-line program that sorts a DOS file, plots the results as a histogram, calculates mean and variance, uses a full-screen editor to create a memo combining histogram, statistics, and descriptive test, issues a DOS command to the PC, dials a host computer, and ships off the memo . . . not bad for 11 lines. A built-in terminal emulator can be toggled on/off with a function key so that executing programs on the host or PC can continue executing. The simulated terminal mode is the extended Datamedia 1520, but some customization is possible.

Both full-screen and del editors are available. The full-screen editor is nicely done, but on my computer and graphics card (both Columbia), I must operate with the session parameter  $E=64$  to reduce the small band of

dashes that flashes randomly when any key is hit. This is supposed to disable the full-screen editor, but in fact it still works. The only degradation in performance is that the editor does not exit in a way to restore the original screen: line numbers and a reverse video EDITIN are left and they do not scroll off screen. The full-screen editor works fine if I am willing to tolerate snow for each key press; alternately, I could define a function key to clear the screen following exit from the full-screen editor. These are relatively minor inconveniences, but you may feel that for \$595 you should get all features of the "supported" hardware.

TOOLS I contains a large collection of special functions for maintaining other functions and workspaces, transferring a workspace to a file and back, and documenting workspaces. Most of these are of interest only to the professional applications developer.

APL★PLUS/PC allows interfaces to non-APL programs such as assembler or DOS and allows  $\square$  PEEK- and  $\square$  POKEing. The only documentation is in the reference manual and is difficult to follow; reportedly STSC has improved this in Version 4.0.

Both a logical set of APL file system commands (compared to the awkward IBM shared variables) and an interface to "native" DOS files are provided. Up to 20 files can be tied at one time. There is complete forward compatibility with APL files, but Versions 2.0 or 2.6 will not work on Version 3.1 files.

A set of powerful formatting, error trapping, and string searching primitives is supplied. The  $\square$  FMT, adapted from the mainframe APL★PLUS, is particularly well suited to business report formatting. Some additional functions in a workspace help you set up tables, and even more functions are available in TOOLS I, including functions that can direct output to screen, APL file, DOS file, or printer and provide pagination, page breaks, and so on.

APL★PLUS/PC has 135 system functions and over 200 utility programs. About 26 of these are also system commands of the )xxx type long familiar to APL users, but most of the system functions are usable within functions. In Version 3.1, )SAVE, )LOAD, and )COPY can be painfully slow, with lots of thrashing about by the disk drives;

)LOAD and )COPY have been speeded up in Version 4.0. An alternative is to use a RAM disk—for safety, instead of using )OFF, you can write a small SIGNOFF function that will copy files/workspaces to permanent storage and exit via  $\square$  SA OFF.

### Enhancements

Many of the enhancements to APL, especially adaptation to the PC environment, come in the form of special system-level functions, variables, and constants. Because they are system-level features, they are always available in every workspace. These are largely, but not entirely, unique to each vendor, and any APL program that incorporates any of the nonstandard system features will not be compatible with other products. However, because STSC provides the Run Time System as a total environment for applications, this is of little concern. It would be a problem only for an individual trying to port applications between a mainframe (even between some STSC products) and APL★PLUS/PC or between friends with different systems.

The 135 system features in the APL★PLUS/PC system provide information about the session, active work-

space, and objects in the workspace; retrieve objects or entire workspaces from disk; assist in debugging; communicate with other devices; manipulate screen (cursor, window, graphics, and so on) and sound effects; and observe and manipulate the operating environment.

System variables contain information about the environment, such as comparison tolerance, index origin, printing precision, and random link, about latent expressions, HELP filenames, what action to take if execution stops for immediate input, and memory segment for peek/poke. Other system variables are for cursor position, keyword setting, printing width, and screen window control.

System constants comprise a dozen constants that control the character set (atomic vector) and generate bell, backspace, delete, escape, form/line feed, new line feed, null character, and so on.

Table I (below) gives an indication of the depth of support.

### Benchmarks

An excellent comparison of STSC APL★PLUS/PC (Version 2.6) and IBM APL (Version 1.00) appeared in the March

|   |
|---|
| Files: 21 for APL and 13 for DOS                          |
| Character constants: 9                                    |
| Debugging tools: 8  |
| Detached I/O: 4   |
| Disk: 4   |
| DOS: 1  |
| Exception handling: 5                                     |
| Execution: 6  |
| Function Definition: 11                                   |
| General Information: 12                                   |
| Graphics/Screen: 21                                       |
| Help: 3   |
| Input/Output: 10  |
| Keyboard: 6   |
| Latent Expression: 3                                      |
| Machine Language: 3 (call/interrupt/symbol table pointer) |
| Memory: 3 (peek/poke/seg)                                 |
| Object: 9   |
| Session: 4  |
| State Indicator: 6  |
| Workspace/Object Manipulation: 12                         |
| Workspace: 9  |
| Other: 1 substring searching, 1 delay execution           |

Table I  
List of System Features by Type

1984 issue of *Byte*. The only weakness of these comparisons was that the exact timing algorithm was not stated. I have used the same benchmarks but have replaced J. Bensimon's large chess problem with the *DDJ* Savage floating point benchmark and have used the documented *TIMER* function supplied by STSC in the workspace *CIN0* (Listings One – Four, page 98). This *TIMER* corrects for all overhead.

All benchmarks were run first on my Columbia MPC 1600-1 without an 8087. To include comparisons with the earlier version of STSC APL\*PLUS/PC used in the *Byte* article (Version 2.6) and to obtain 8087 data, I enlisted the unsuspecting help of the STSC small systems hotline staff who happily performed tests No. 1 – 19 on both Versions 2.6 and 3.1, both with and without the 8087 (a *POKE* turns the 8087 off, and the machine acts as if it were not there in all respects).

The comparable STSC benchmarks were all uniformly faster than mine; this precipitated a number of phone calls to both Columbia and STSC and between the two vendors. We could not discover any obvious reason for this discrepancy, outside of a difference in the way the benchmarks were run: the STSC benchmarks were run using a master program to feed the arguments to *TIMER*, while my benchmarks were originally performed manually. The STSC benchmarks for No. 5 were performed on the wrong variable, so I have omitted No. 5, but the STSC benchmark No. 2 was still about half the value that I obtained on the Columbia.

I had originally decided to report only ratios for the STSC data, but I later decided to check the performance on an IBM myself. With the cooperation of two colleagues at work, I was able to boot my exact workspace on an IBM XT with 640K RAM and to perform all benchmarks in the same manner as on my Columbia at home. The XT without 8087 gave much closer agreement to the Columbia values, including a value of about 9 msec for No. 2.

The disagreement with STSC's values still bugged me, so I coded the master program and reran all benchmarks on my Columbia. The discrepancies were still there: I then copied all variables and functions into a )CLEAR workspace and reran all benchmarks.

Lo and behold, the value of 4.5 flashed up for No. 2, in close agreement with STSC's values; all other values improved significantly in agreement.

An inexplicable bug apparently had crept into the original workspace and affected timing, not accuracy, in a manner not due to pending functions, limited  $\square WA$ , or any other obvious problem; I have shipped a copy of the "good" and "bad" workspaces off to STSC for examination. Benchmarks in APL are not as reproducible as they may be in other languages because a lot of bookkeeping goes on "behind closed doors." The state of the workspace depends to some extent on what has preceded it. Therefore, I now store a clean workspace for benchmarking and )LOAD it fresh for each run.

First, let's look at the performance of Version 2.6 (the *Byte* review version) compared with the current Version 3.1 without an 8087. Curiously, some of the simple functions have gotten slightly worse (about 10% worse for No. 1 plus reduction, No. 3 maximum reduction, No. 6 indexing, No. 8 take, and No. 10 transposition); however, much more substantial improvements were realized in No. 14 matrix division (66%), No. 15 Fibonacci series (33%), and No. 17 division (14%). Comparing the two versions with an 8087 shows improvements in the newer version of 69% for No. 1 plus reduction and 87% for No. 14 matrix division, with smaller improvements in Nos. 2, 13, 15, 16, 17, and 19. There are 10 – 15% decrements in Nos. 3, 6, and 10.

Use of the 8087 math coprocessor gives improvements of 100 – 3600% in some benchmarks (Version 3.1; Nos. 1, 4, 13, 14, 16, 17, 18 and 19), particularly the logarithm-exponential and transcendental benchmarks.

A comparison of the Columbia MPC 1600-1 benchmarks with those for the IBM XT, both without 8087, shows the Columbia (256K with  $\square WA$  of 95K) running fairly consistently at about 96% of the speed of the XT (640K with  $\square WA$  of 469K). There are insignificant differences between the VP and MPC (Cols. E+H) and between the XT and PC (Cols. C+F and D+G), showing that  $\square WA$  is not an important variable.

The well-known Sieve of Eratosthenes (No. 20) involves a lot of branch-

ing and showed little enhancement using the 8087, as might be expected. The Savage floating-point benchmark (No. 21) can be coded easily and elegantly in APL as shown in Listing Three. The first line prints a time-stamp for benchmarking, initializes two variables, and precomputes the branchpoints for the iteration. The second line evaluates the expression and tests for iteration number. The last line prints a second time-stamp. Without an 8087 coprocessor (which significantly enhances exactly the primitives used in this calculation), the speed was about equal to double-precision BASIC, 855 seconds on the Columbia and 833 seconds on the XT, but there was a low error of 1E-9. With the 8087, the time was only 198 seconds, which is not bad for an interpreted language. This compares favorably with several compiled languages (e.g., Supersoft Fortran on the PC), and is slower but more accurate than the only other APL listed, IBM APL (with 8087 implied), but I have not seen the coding used.

Users usually base their choices between upgrading to newer versions or between different vendors more on features than on small differences in "speed" benchmarks. However, STSC has clearly succeeded (intentionally or not) in reducing any glaring weaknesses that the *Byte* comparison might have showcased, particularly the laggard performance of matrix division on the older Version 2.6.

Finally, some minor quibbles: I have used the *Byte* benchmarks for consistency, although some of these (Nos. 14 and 19) involve two primitives when a new variable might be more appropriate. The PRIMES benchmark was from the *Byte* article; the variables should have been localized in the header.

## Development Package?

What kinds of applications would you develop in APL? Surprisingly, the main revenue stream of the commercial APL systems is business and not scientific applications. For example, APL did not come to our IBM 3081 mainframe because the scientists or statisticians wanted it; it came because a tie-wearing manager type wanted *ADRS*, which just happens to be written in APL. Two examples of applications from the recent STSC PLUS\*

NEWS newsletter are a financial system and a DBMS for the broadcasting industry.

R.W. Butterworth makes some interesting points—albeit directed mainly toward mainframe systems—in his article “When APL is Inappropriate” (in *APL in Practice*, page 43). The main criterion is who or what reads the program most often or importantly during its full life cycle. Low-level languages (e.g., assembly code or even C) favor machine readability, while very high-level languages favor people readability. Therefore, general support utilities, high-volume processing, or very complex calculations (“number crunching” scientific applications) are not good for APL because they are too machine intensive. On-line real-time applications involving complex tasks are also not good: portability is somewhat limited if the code involves lots of enhancements, but this is true of most high-level languages. Hybrid solutions, however, offer many benefits; APL★PLUS/PC offers interfaces to machine code and communications.

When is development on micros using APL warranted? Any project that involves a lot of *ad hoc* analysis is a good candidate for APL. However, with the Run Time System, all possible *ad hoc* variants must be predetermined, and either menu choices or pre-defined function names are required to determine which one will be used. This is probably not a limitation. If the environment cannot be predefined, you have only two choices. Either your APL application using the Run Time System must have a command simulation processor to allow the user to enter equations, possibly with the APL keyword system, or you should suggest the purchase of the full APL system. You can then supply a more “standard” set of functions and perhaps some training in APL.

Probably the most useful application of the APL Run Time System would be situations where the package needs to be customized quickly or frequently in a manner that cannot be achieved by menu choices of options, input of formulae, and so on, or if your business consists of lots of small custom jobs. If these concepts are correct, then I find the pricing of the Run Time System to be out of whack because it seems to pre-

# Debugging Bugging You?

Torpedo program crashes and debugging delays with debugging dynamite for the IBM PC ...

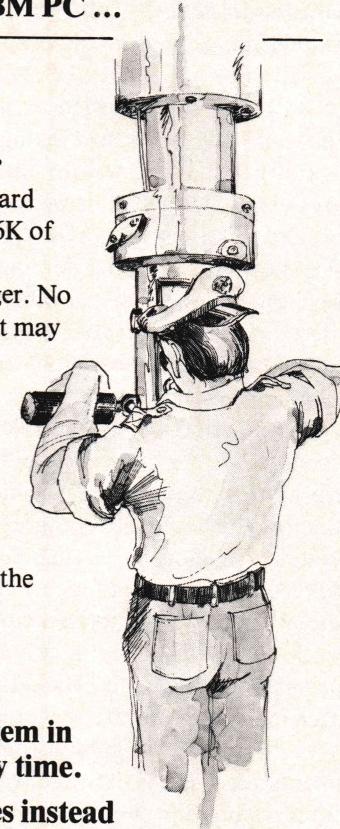
## UP PERISCOPE!

### First, you install the hardware.

The hardware's a special memory board that fits in a PC expansion slot. Its 16K of write-protected memory contains Periscope's resident symbolic debugger. No runaway program, however berserk it may be, can touch this memory!

### Then you UP PERISCOPE.

Use Periscope's push-button break-out switch to interrupt a running program ... even when the system's hung! Periscope supports Assembly, BASIC, C and Pascal. In addition to the usual debugging capabilities, some of Periscope's features are:



**Stop your system in its tracks at any time.**

**Use symbol names instead of addresses.**

**Run a program on one monitor and debug on another.**

**Monitor your program's execution with Periscope's comprehensive breakpoints.**

**Debug memory-resident programs.**

### Put your time to better use.

The Periscope system is \$295. It carries a 30-day money-back guarantee and includes the memory board, remote break-out switch, debugger software, 100-page manual, and quick-reference card. The memory board is warranted for one year. A demonstration disk is \$5.00.

System requirements for Periscope are an IBM PC, XT or Compaq, PC-DOS, 64K RAM, 1 disk drive and an 80-column monitor. For MasterCard and Visa orders only, call 800/421-5300 (ext. R96) 24 hours a day. For additional information, call 404/256-3860 from 9 AM to 5 PM Eastern Time.

Get your programs up and running;

**UP PERISCOPE!**

Data Base Decisions / 14 Bonnie Lane / Atlanta, GA 30328

Circle no. 30 on reader service card.

clude low-volume projects. (Perhaps a better pricing scheme for STSC would be a higher initial purchase price with possibly a small royalty fee.)

### Programming in APL\*PLUS/PC

To illustrate some of the fun and problems involved in using APL\*PLUS/PC, I decided to relate my experiences in converting a BASIC program (Listing Five, page 99) that I had written into the APL\*PLUS/PC system (Listing Six, page 101 and Figure page 100). My Columbia came with Perfect Filer, one of the weakest members of the otherwise quite good Perfect series. Perfect Filer does not have math capabilities, so to do a billing system for my wife's small private practice, I used the Filer simply to store the raw data. The bills then were written to a native DOS file (a:newaku.mss). The BASIC program read through this file line by line (or record by record) and looked for certain keywords/symbols. It then totaled current month, added balance forward, and subtracted any payments. The total was written in the appropriate spot, and if the balance forward was greater than the payments received, a message was written that payment in full is required each month; the revised bill for the first client was written out (a:new0ak.mss) then on to the next client.

Although processing the file line by line is probably the most natural way for a BASIC programmer to attack this problem (and was the approach I originally took before I had APL\*PLUS/PC), an APL programmer knows that you really want to process all records simultaneously. The only problem is to differentiate between the clients. The APL program took about six hours compared to about twenty hours for the original BASIC program. One of the nicest features of APL\*PLUS/PC is the full-screen editor and scrolling buffer. These allow you to try many algorithms easily; for example, you can use calculator mode to run-modify-run-modify and so on by editing on the screen. Then, at some point, you can write a function and further modify it using the full-screen editor )EDIT.

A native DOS file is simply a long string of bytes in APL (8501 in this example), so it is easy to search the string for all occurrences of the keywords/symbols (variable ALL gives the posi-

tions). Originally I did this with a series of  $\Box SS$  substring searches. The actual numbers were then a fixed number of bytes away from these locations (line [6] of *PROCESS*). A function (*CLEAN*) was used to clean up any stray characters, to convert the normal minus to an APL negative, and to insert 000 in blank "numeric" fields. The characters then were converted to numbers. The resulting vector CM was a list of all session charges in order.

How to sum by client? There are usually several ways to do almost anything in APL. Much of what follows could have been avoided by the use of extended operators in APL2 or even a loop with the take primitive. However, the method that I found using APL\*PLUS/PC is more typical. Line [7] of *PROCESS* reshaped the vector into a matrix that had number columns = number of clients (4 here) and number rows = total number of sessions (12; matrix A). Each column contained the original vector of session fees. What was then needed was to discover a mask of 0,1's that could be used to multiply this matrix so that the columns (clients) could be summed to give the current month's total charges for each client.

Because of the layout of the bill, the location of the balance forwards always occurred right after the current month's charges. Therefore, a matrix formed by finding the location of all keywords/symbols less than the balance forwards (matrix B) would identify each client—the first client would have all columns filled, the second n-1 columns, and so on to the last client, which would have only the last column filled. The last step would be to keep only the first 1 for each client; this was accomplished using the "less than scan" (matrix C). The final expression gives the sums CSUMS.

The program actually ran rather slowly. This was obviously due to the multiple string searches using  $\Box SS$  (reportedly speeded up in Version 4.0). It would be nicer and faster to search the main string only once, at least for the new sessions if not for everything. This proved possible by using the membership function (see *FINDALL*) to find all occurrences of the symbols i, g, A, and W (for individual, group, administrative, and workshop). This would include any words that happened to have these

letters present, but the actual location would have a blank on either side. Therefore, all you had to do was examine the locations to the left and right, test for blanks, and do a logical AND (line [5] of *PROCESS*). The resulting algorithm was twice as fast as using the multiple  $\Box SS$ 's.

The APL version contains about the same number of lines—excluding comments and function headers—but runs almost exactly three times faster! Further optimization is undoubtedly possible.

Having the APL\*PLUS/PC system in the first place would have made it unnecessary to improve Perfect Filer because the proper data base could have been set up easily. However, the example illustrates many of the features of programming in APL\*PLUS/PC: it is fast, fun, well-adapted to the PC environment, and encourages elegant global solutions. Of course, what may be elegant for me may be pedestrian for another APL user.

APL was designed as a heuristic tool before it became a computer language. I find describing the conceptualization in English much more difficult than the symbolic description in APL shown in Listing Six. Unless you have some experience with APL, you may find the opposite to be true. Those with some APL under their belt tend to program in a twilight zone between the English verbalization and the vector/matrix concepts of APL. In many instances, you know that there is a trick; you just have to remember it, probably by fiddling on screen. In this case, I knew there was a scan function that would do it—line [7]—I just couldn't remember which one.

### Menu a la TOOLS

Having received the TOOLS package just as I was completing the major portion of this review, I have not had a lot of time to fully evaluate all of its features. I have spent most of my time with the menu and screen design sections.

The benchmarks table (Table II, page 95) was produced using functions in the distribution workspace *FORMAT* and *OUTPUT* workspace from TOOLS. Due to scanty documentation, most of the minor difficulties involved function *COLNAMES* from *FORMAT*, which required a bit of fussing to work

| Computer                     |                     | A    | B    | C    | D    | E    | F    | G    | H    |
|------------------------------|---------------------|------|------|------|------|------|------|------|------|
|                              |                     | PC   | PC   | PC   | PC   | VP   | XT   | XT   | MPC  |
| APL Version                  |                     | 2.6  | 2.6  | 3.1  | 3.1  | 3.2  | 3.1  | 3.1  | 3.1  |
| 8087?                        |                     | 8087 |      | 8087 |      |      | 8087 |      |      |
| OWA                          |                     | 410K | 410K | 302K | 302K | 87K  | 480K | 480K | 95K  |
| RAM                          |                     | 512  | 512  | 512  | 512  | 512  | 640  | 640  | 256  |
| 1. PLUS REDUCTION            | Z←+/VI              | 104  | 158  | 32   | 169  | 174  | 32   | 169  | 175  |
| 2. LOGICAL REDUCTION         | Z←∨/VL              | 5    | 6    | 4    | 4    | 5    | 5    | 4    | 5    |
| 3. MAXIMUM REDUCTION         | Z←⌈/I]MI            | 30   | 30   | 35   | 34   | 35   | 35   | 35   | 36   |
| 4. EXPONENTIATION            | Z←VI*.1             | 1697 | 9571 | 1704 | 9249 | 9426 | 1704 | 9250 | 9473 |
| 5. ABSOLUTE VALUE            | Z← VR               |      |      |      |      |      | 130  | 65   | 120  |
| 6. INDEXING                  | Z←VR[VI[1..20]]     | 22   | 23   | 26   | 26   | 26   | 25   | 25   | 26   |
| 7. SORTING                   | Z←VI[⍋VI]           | 118  | 118  | 119  | 119  | 125  | 116  | 116  | 122  |
| 8. TAKE                      | Z←2 1↑MR            | 29   | 29   | 29   | 30   | 31   | 29   | 30   | 31   |
| 9. MEMBERSHIP                | Z←VI∊VI             | 149  | 149  | 154  | 153  | 154  | 154  | 153  | 156  |
| 10. TRANSPOSITION            | Z←2 1@MC            | 64   | 64   | 70   | 70   | 71   | 71   | 71   | 72   |
| 11. OUTER PRODUCT, CHARACTER | Z←VC..=VC           | 131  | 130  | 137  | 137  | 143  | 136  | 136  | 143  |
| 12. OUTER PRODUCT, INTEGER   | Z←(1..50)∘.+1..50   | 445  | 445  | 440  | 441  | 465  | 449  | 444  | 467  |
| 13. INNER PRODUCT, REAL      | Z←VRL.+VR           | 346  | 553  | 280  | 575  | 591  | 280  | 575  | 594  |
| 14. MATRIX DIVISION          | Z←MR←10↑VR          | 1486 | 2217 | 196  | 755  | 778  | 196  | 755  | 782  |
| 15. FIBONACCI SERIES         | Z←FBNCC (LISTING 1) | 3832 | 3945 | 2487 | 2687 | 2801 | 2435 | 2638 | 2781 |
| 16. MULTIPLICATION           | Z←VR×3.14           | 146  | 478  | 112  | 490  | 507  | 113  | 489  | 509  |
| 17. DIVISION                 | Z←VR÷3.14           | 152  | 733  | 118  | 632  | 652  | 118  | 632  | 657  |
| 18. LOGARITHM                | Z←@VR               | 131  | 5085 | 131  | 4872 | 4970 | 131  | 4872 | 4994 |
| 19. SINE                     | Z←10↑VR×.1          | 484  | 6060 | 453  | 6103 | 6224 | 444  | 6094 | 6243 |
| 20. SIEVE OF ERATOSTHENES    | (LISTING 2) SEC.×10 |      |      |      |      |      | 1634 | 1666 | 1755 |
| 21. SAVAGE FLOATING PT.      | (LISTING 3) SEC.    |      |      |      |      |      | 198  | 833  | 855  |

1) Times are in milliseconds per execution for Benchmarks #1-19 and seconds for #20-21. #20 is reported for 10 executions, as per convention.

2) TIMER (APL\*PLUS/PC Workspace CINO, see Listing 4) was with 100 repetitions and with 'R←0' for NULLEXP, except for #15 where 'R←NONFN' was used. NONFN is niladic and simply assigns R←0. Benchmarks #20 and #21, since they are full functions and are relatively slow, were timed with DAI[2], as shown in the Listings 2-3.

3) The PC and VP had monochrome monitors, the XT had both monochrome and color monitors and boards and the MPC had color monitor. All tests were done in monochrome mode. The PC, VP and XT used DOS 2.1, the MPC 2.0.

4) The variables, as in the Byte March 1984 review, are:

```

MI←10 10↑VI←(500↑0 1 0 0 1)/1..500
VL←1 0 1 1 0 0 0 1
MR←10 10↑VR←VI+0.1
MC←26 26↑VC←'ABCDEFGHIJKLMNPQRSTUVWXYZ'

```

Table II

properly. Use of the functions in *OUT-PUT* was relatively straightforward.

To test menu and screen functions from *TOOLS*, I decided to adapt some recombinant DNA programs to the PC using these features. A menu is divided into title, text, choice, and footnote sections and allows multiple choices with validity checking; it has one write-in area, time-out features, and full color/monochrome decorations. An enormous number of possible options are available. A screen is similar but more general; for example, multiple write-ins can be made. Some run-of-the-mill screens can be set up very simply: 'MENU MENU' CHOICE FROM '/AU

*CHAMBERTIN/PEARL RIVER SEA-FOOD/CHUCK E. CHEESE'.*

In setting up the main DNA menu, I decided it would be useful to have the name of the registered owner appear somewhere. It was not at all obvious how to enter current information such as date, time, or user into the menu. The screen is first designed and then "compiled" for fast display (not a true compilation, rather an interpretation into more primitive APL functions). Several of the key functions are locked and the terse error messages are not very helpful. The documentation is weak on how to create certain effects; some of these can be gleaned only from

the single example in the workspace (for example, how to test for color boards and change attributes appropriately or how to center text). After a few calls to the hotline, I found that the only way to enter current information is to use some special editing functions to replace defined fields in the "compiled" menu.

Once the menu is "compiled," it flashes instantly on the screen—unless you are using E=64 at sign-on, in which case it takes several seconds to paint the screen. Despite the shortcomings of the documentation, once you get the hang of it, you can design some very nice menus.

I also had several problems with the screen functions and documentation. In the step-by-step discussion of how to design a sample screen, a critical "press enter" was omitted following the naming of the first field (*EXT*). No other exiting gambit works. The documentation is also confusing concerning the use of the function *SCFKEYSOFF* with *SCCLOSE*; you must not use these together. Instead, if you want to set function keys, you must follow a more complicated procedure that is outlined in the documentation, but the incompatibility is not made clear.

*SCINIT* calls the undocumented *SCCURSORSET*, which is  *IO* sensitive (=0 works), a function similar in purpose to *WHITEWASH* in the *MENU* workspace for color/mono conversions. If you want to re-use a screen (as I did), you must blank out any filled-in fields that you do not want to carry forward before you store it using *SCOUT-PUT*'. After finally working out all of these details, I had a screen that looked nice, only lit up the write-in field when a function key was pressed and waited for proper input; unfortunately, even when compiled the screen took about three seconds to appear, much slower than a menu. The *SCREEN* workspace was the least satisfactory of the *TOOL* features that I was able to test.

#### You Get What You Pay For

STSC APL\*PLUS/PC is not a cheap package, but it's worth every penny you spend on it. The claim is that the package is a development tool... and it is. It's possible to develop serious packages or to use the system in conjunction with LANs or mainframes to

## MEMO: C Programmers

# QUIT WORKING SO HARD.

These people have quit working so hard: IBM, Honeywell, Control Data, GE, Lotus, Hospitals, Universities & Government Aerospace.

## THE GREENLEAF FUNCTIONS™

THE library of C FUNCTIONS that probably has just what you need... TODAY!

- ... already has what you're working to re-invent
- ... already has over 200 functions for the IBM PC, XT, AT, and compatibles
- ... already complete... already tested... on the shelf
- ... already has demo programs and source code
- ... already compatible with all popular compilers
- ... already supports all memory models, DOS 1.1, 2.0, 2.1
- ... already optimized (parts in assembler) for speed and density
- ... already in use by thousands of customers worldwide
- ... already available from stock (your dealer probably has it)
- ... It's called the **GREENLEAF FUNCTIONS**.

Sorry you didn't know this sooner? Just order a copy and then take a break — we did the hard work. Already.

**THE GREENLEAF FUNCTIONS GENERAL LIBRARY:** Over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and system interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Poll mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you. From Greenleaf Software.

... **Specify compiler** when ordering. Add \$7.00 each for UPS second-day air. MasterCard, VISA, check, or P.O.

|                        |                              |
|------------------------|------------------------------|
| ◆ Compilers:           | ◆ General Libraries....\$175 |
| CI C86.....\$349       | (Lattice, Microsoft, Mark    |
| Lattice .....\$395     | Williams, CI C86)            |
| Mark Williams ...\$475 | ◆ DeSmet C.....\$150         |
|                        | ◆ Comm Library.....\$160     |



**GREENLEAF SOFTWARE, INC.**

2101 HICKORY DRIVE • CARROLLTON, TX 75006 • (214) 446-8641

Circle no. 43 on reader service card.

# DDJ BACK ISSUES

## #68 Volume VII, No. 6:

Multi-68000 Personal Computer—PDP-1802, Part One—Improved LET for LLL Basic—CP/M Print Utility.

## #69 Volume VII, No. 7:

IBM-PC Issue: CP/M-86 vs. MSDOS (A Technical Comparison)—Hi-Res Graphics on the IBM-PC—PDP-1802, Part II—Review of Word Processors for IBM.

## #70 Volume VII, No. 8:

Argum "C" Command Line Processor—SEND/RECEIVE File Transfer Utilities—Intel's 8087 Performance Evaluation.

## #71 Volume VII, No. 9:

FORTH Issue: Floating-Point Package—H-19 Screen Editor—Relocating, Linking Loader—Z8000 Forth—Forth Programming Style—8086 ASCII-Binary Conversion Routines—CP/M Conditional SUBMIT.

## #72 Volume VII, No. 10:

Portable Pidgin for Z80—68000 Cross Assembler, Part I—MODEM and RCP/MS—Simplified 68000 Mnemonics—Nested Submits—8086/88 Trig Lookup.

## #73 Volume VII, No. 11:

Wildcard UNIX Filenames—Tests for Pidgin—68000 Cross Assembler Listing, Part 2—Adding More BDOS Calls—The Perfect Hash—BASIC Memory Management—Benchmarks for CP/M-86 vs. MSDOS, and the 8087.

## #76 Volume VIII, Issue 2:

PISTOL, A Forth-like Portably Implemented Stack Oriented Language—Program Linkage by Coroutines. Forth to BASIC—Linking CP/M Functions to Your High-Level Program—Concurrent CP/M-86—CP/M-80 Expansion Card for the Victor 9000—REVAS Disassembler.

## #77 Volume VIII, Issue 3:

The Augusta P-Code Interpreter—A Small-C Operating System—6809 Threaded Code: Parametrization and Transfer of Control—A Common-Sense Guide to Faster, Small BASIC—A Fundamental Mistake in Compiler Design—Basic Disk I/O, Part I.

## #78 Volume VIII, Issue 4:

RECLAIM Destroyed Directories—Binary Magic Numbers—8080 Fig-Forth Directory & File System—“SAY” Forth Votrax Driver—TRS-80 8080 to Z80 Translator—Basic Disk I/O, Part II.

## #79 Volume VIII, No. 5:

The Augusta Compiler—A Fast Circle Routine—Enhancing the C Screen Editor—Shifts and Rotations on the Z80—The SCB, TSX, and TXS Instructions of the 6502 and 6800—MS-DOS vs. CP/M-86—Controlling MBASIC—The Buffered Keyboard—IBM PC Character Set Linker—Flip Utility for the IBM PC.

## #80 Volume VIII, Issue 6:

Fast Divisibility Algorithms—B-Tree ISAM Concepts—CP/M BDOS AND BIOS Calls for C—Serial Expansion in Forth—Fast Matrix Operations in Forth, Part I—Yes, You Can Trace Through BDOS—Julian Dates for Microcomputers—8088 Addressing Modes—8088 Line Generator—CP/M Plus.

## #81 Volume VIII, Issue 7:

The Augusta Compiler, continued—RED: A Better Screen Editor, Part I—Anatomy of a Digital Vector and Curve Generator—Fast Matrix Operations in Forth, Part II—The AGGHHHI Program—MBOOT Revisited—CP/M Plus Feedback—MS-DOS Rebuttal—68000 Tools—Sizing Memory on the IBM PC.

## #82 Volume VIII, Issue 8:

Serial-to-Parallel: A Flexible Utility Box—McWORDER: A Tiny Text Editor—And Still More Fifth Generation Computers—Specialist Symbols and I/O Benchmarks for CP/M Plus—CP/M Plus Memory Management—Zero Length File Test—PAUSEIF, QUITIF, and now SKIPIF—ACTxx Cross Assemblers.

## #83 Volume VIII, No. 9:

FORTH ISSUE: Forth and the Motorola 68000—Nondeterministic Control Words in Forth—A 68000 Forth Assembler—GO in Forth—Precompiled Forth Modules—Signed Integer Division—Some Forth Coding Standards—The Forth Sort.

## #84 Volume VIII, No. 10:

Unix to CP/M Floppy Disk File Conversion—A Small-C Help Facility—Attaching a Winchester Hard Disk to the S-100 Bus—Using Epson Bit-Plot Graphics—8086/88 Function Macros—Auto Disk Format Selection—CP/M Plus Device Tables.

## #85 Volume VIII, Issue 11:

A Kernel for the MC68000—A DML Parser—Towards a More Writable Forth Syntax—Simple Graphics for Printer—Floating-Point Benchmarks.

## #86 Volume VIII, Issue 12:

Faster Circles for Apples—Cursor Control for Dumb Terminals—Dysan's Digital Diagnostic Diskette—Interfacing a Hard Disk Within a CP/M Environment—The New MS-DOS EXEC Function.

## #87 Volume IX, Issue 1:

A Structured Preprocessor for MBASIC—A Simple Window Package—Forth to PC-DOS Interface—Sorted Diskette Directory Listing for the IBM PC—Emulate WordStar on TOPS-20—More on optimizing compilers—The PIP mystery device contest.

## #88 Volume IX, Issue 2:

Telecommunications Issue: Micro to Mainframe Connection—Communications Protocols—Unix to Unix Network Utilities—VPC: A Virtual Personal Computer for Networks—PABX and the Personal Computer—BASIC Language Telecommunications Programming—U.S. Robotics S-100 Card Modem.

## #89 Volume IX, Issue 3:

RSA: A Public Key Cryptography System, Part I—Introduction to PL/C: Programming Language for Compilers—Program Design Using Pseudocode—More on Binary Magic Numbers—How fast is CP/M Plus?—CP/M 2.2 BIOS Function: SELDSK—The results of the Floating-Point benchmark.

## #90 Volume IX, Issue 4:

Optimizing Strings in C—Expert Systems and the Weather—RSA: A Public Key Cryptography System, Part II—Several items on CP/M Plus, CP/M v2.2 Compatibility—BDOS Function 10: Vastly Improved—More on MS-DOS EXEC Function—Low-Level Input-Output in C.

## #91 Volume IX, Issue 5:

Introduction to Modula-2 for Pascal Programmers—Converting Fig-Forth to Forth-83—Sixth Generation Computers—A New Library for Small-C—Solutions to Quirks in dBASE II.

## #92 Volume IX, Issue 6:

CP/M on the Commodore 64—dBASE II Programming Techniques—First Chinese Forth: A Double-Headed Approach—cc-A Driver for a Small-C Programming System—A New Library for Small-C (Part II)—Comments on Sixth Generation Computers—Review of Turbo Pascal.

## #93 Volume IX, Issue 7:

RSX under CP/M Plus—p-A Small-C Preprocessor—A Simple Minimax Algorithm—Languages and Parentheses (A Suggestion for Forth-like Languages)—Comments on assembly language development packages, RSX to patch CP/M 2.2 with CP/M, iRMIS-86 for the IBM PC, C Programming Tools.

## #94 Volume IX, Issue 8:

SCISTAR: Greek and Math Symbols with WordStar—A File Comparator for CP/M Plus—Designing a File Encryption System—A Small-C Concordance Generator.

## #95 Volume IX, Issue 9:

Forth Special Issue!—File Maintenance in Forth—Forth and the Fast Fourier Transform—Computing with Streams—A Forth Native-Code Cross Compiler for the MC68000—The FVG Standard Floating-Point Extension—CP/M Plus: Interbank Memory Moves Without DMA—ways to make C more powerful and flexible.

## #96 Volume IX, Issue 10:

More dBASE II Programming Techniques—Simple Calculations with Complex Numbers—GREP C: A Unix-like Generalized Regular Expression Parser—An optimization scheme for compilers, MSDOS 2.0 Filters, Sizing RAM under MSDOS, Two programming systems illustrating Runge-Kutta integration.

## #97 Volume IX, Issue 11:

Adding Primitive I/O Functions to mulISP—Program Monitor Package: Using Interrupts to Instrument Applications—CP/M 2.2 Goes PUBLIC—A Guide to Resources for the C Programmer—RESORT.

## #98 Volume IX, Issue 12:

Varieties of Unix—Unix Device Drivers—A Unix Internals Bibliography—A file Browser Program—An Introduction to Parsing.

## TO ORDER:

Send \$3.50 per issue to: **Dr. Dobb's Journal**, 2464 Embarcadero Way, Palo Alto, CA 94303

Please send me the issue(s) circled: **66 68 69 70 71 72**

**73 76 77 78 79 80 81 82 83 84 85 86**

**87 88 89 90 91 92 93 94 95 96 97 98**

I enclose \$\_\_\_\_\_ (U.S. check or money order).

Outside the U.S., add \$.50 per issue.

I have read the postal instructions and understand that I will not receive my order unless I have sent the correct payment amount.

Please charge my:  Visa  M/C  Amer. Exp.

Card No. \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Availability on first come/first serve basis. Outside the U.S. add \$.50 per issue ordered. Price includes issue, handling, and shipment by second class or foreign surface mail. Within the U.S., please allow 6-9 weeks to process your order second class. For faster service within the U.S., we'll ship UPS if you add \$1.00 for 1-2 issues and \$.50 for each issue thereafter. We need a street address, not a P.O. Box. Airmail rates: To Canada add \$1.75 per magazine, all other foreign add \$3.00 per magazine.

Circle no. 81 on reader service card.

do serious work (for example, work done in one of our statistics groups). Certain low-volume applications may not have a low price tag (because of the \$50-\$100 Run Time System license fee), but this may change.

I find the most curious aspect of APL to be its marketing by STSC, IBM, and others. STSC seems to be following IBM's lead . . . or non-lead. Although STSC runs occasional full-page ads in

the computer rags, there is a minimum of second sourcing. For example, I have seen STSC APL★PLUS/PC offered from only two alternate vendors. One is a well-known programmers shop (at about a 16% discount), and the other is a vendor specializing in 8087 chips and related software. Of course, IBM APL is never advertised in the usual computer rags and is never second sourced (I believe). Apparently

a lot of work gets done at IBM in APL, but the marketing mavens believe in benign neglect for this product. Why STSC is not more vigorous in promoting its PC version will probably remain a mystery. The new Pocket APL may be an attempt to popularize their excellent products; if so, time will tell.

DDJ

(More reviews on page 102)

## Software Reviews (Text begins on page 88)

### *Listing One*

```
▽ Z←FBNCC
[1]   Z← 1 1
[2]   L:→(100)⍳Z←Z,+/↑2↑Z)/L
▽
```

```
▽ R←NONFN
[1]   R←0
▽
```

End Listing One

### *Listing Two*

```
▽ PRIMES
[1]   DAI[2] ⋄ I←1 ⋄ F←(8191⍳Y),E
[2]   Y:F[I+P×L(8191-I)÷P←I+I+I]←N
[3]   N:→F[I←I+1]
[4]   E:(⍟Y+.=F),' PRIMES' ⋄ DAI[2]
▽
```

End Listing Two

### *Listing Three*

```
▽ A←FPBENCH;N;R
[1]   DAI[2] ⋄ A←N←1 ⋄ R←(2499⍳L),E
[2]   L:A←1+3○↑3○*(A×A)*0.5 ⋄ →R[N←N+1]
[3]   E:DAI[2]
▽
```

End Listing Three

### *Listing Four*

```
▽ □□R+□□N TIMER □□E;□□T1;□□T2;□□T3;□□T4
[1]   □ Result □□R is the time in milliseconds to execute □□E □□N times.
[2]   □ NULLEXP is the APL expression whose time is to be subtracted from
[3]   □ the timing. It is typically 'S←0' if □□E begins with 'S←'.
[4]   □□T2←(□□N×1+⍳NULLEXP)⍳NULLEXP,'0' ⋄ □□T4←(□□N×1+⍳□□E)⍳□□E,'0'
[5]   □ □WA on next lines forces a garbage collection.
[6]   □□R+□WA ⋄ □□T1+□TS ⋄ □□T2 ⋄ □□T2+□TS
[7]   □□R+□WA ⋄ □□T3+□TS ⋄ □□T4 ⋄ □□T4+□TS
```

```

[8]  @@T1← 60 60 60 1000 1↑4@T1
[9]  @@T2← 60 60 60 1000 1↑4@T2
[10]  @@T3← 60 60 60 1000 1↑4@T3
[11]  @@T4← 60 60 60 1000 1↑4@T4
[12]  @@R←(@@T4-@@T3)-@@T2-@@T1

```

End Listing Four

## ***Listing Five***

```

90 PRINT TIME$: SUM=0!: LS$=""
100 OPEN "I", #1, "a: newaku.mss"
105 OPEN "O", #2, "a: newOak.mss"
200 LINE INPUT#1, X$: IF EOF(1) THEN 9000 ELSE 201
201 IF X$=CHR$(13) THEN PRINT#2, X$: GOTO 200
204 A=INSTR(X$, ": $"): IF A THEN 1000 ELSE PRINT #2, X$
205 REM: find name
206 N=INSTR(X$, ". R. "): IF N GOTO 209 ELSE 290
209 L=LEN(X$)-(N+4): RN$=RIGHT$(X$, L)
211 L=1+LEN(RN$)-INSTR(RN$, CHR$(77)): NA$=RIGHT$(RN$, L): GOTO 200
290 FP=INSTR(X$, "Payment"): IF FP GOTO 315 ELSE FI=INSTR(X$, "/8")
300 IF FI GOTO 301 ELSE 310
301 R$=RIGHT$(X$, LEN(X$)-27): L$=MID$(X$, 8, 5)+STR$(VAL(R$))
302 SUM=SUM+VAL(R$): LS$=LS$+L$+" ": GOTO 200

```

(Continued on next page)

## **SMALL C FOR IBM-PC**

Small-C Compiler Version  
2.1 for PC-DOS/MS-DOS  
Source Code included  
for Compiler & Library  
New 8086 optimizations  
Rich I/O & Standard Library

\$**40**

### **CBUG SOURCE LEVEL DEBUGGER FOR SMALL C**

Break, Trace, and Change  
variables all on the  
source level  
Source code included

\$**40**

**Datalight**  
11557 8th Ave. N.E.  
Seattle, Washington 98125  
(206) 367-1803

ASM or MASM is required with compiler.  
Include disk size (160k/320k), and DOS version with order.  
VISA & MasterCard accepted. Include card no. & expiration date.  
Washington state residents include 7.9% sales tax.  
IBM-PC & PC-DOS are trademarks of International Business Machines  
MS-DOS is a trademark of Microsoft Corporation.

Circle no. 31 on reader service card.

## **FORTH**

including  
**SOURCE CODE**

Developing your own system? Or, just curious  
about how things work? Either way, SOURCE CODE  
is a must!

**KFORTH** was developed for use in microprocessor-based controllers used by the U.S. Government. It includes CASE statements, a built-in assembler, and CPM file handling. Best of all, you can change it to fit your needs.

**SUPER KFORTH** was developed for increased speed. It uses DIRECT THREADED CODE and is up to 10x faster. Both are written in assembler and can be assembled using ASM.COM. Both generate reentrant and ROMABLE code.

(For use with Z80, 8080, 8085 CPM systems)

FILL OUT COUPON TODAY AND MAIL TO:

DJ

**KIMRICH COMPUTER DESIGNS, INC.**

10404 Patterson Avenue, Richmond, VA 23233 (804) 741-5930

YES! I want SOURCE CODE! Enclosed is my check for:

KFORTH ..... \$39.95     SUPER KFORTH ..... \$79.95  
(In VA add \$1.60 sales tax (4%))    (In VA add \$3.20 sales tax (4%))

My disk format is: (Call for availability of other formats)

8 inch SSSD     Osborne SD     Kaypro  
 5 1/4 inch SSSD     Osborne DD

For VISA or MasterCard orders phone (804) 741-5930.

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Circle no. 53 on reader service card.

# Software Reviews

(Listing Continued, text begins on page 88)

## *Listing Five*

```
310 A=INSTR(X$, "+") : IF A THEN 311 ELSE 315
311 BA=VAL(MID$(X$, A+2, 5)) : SUM=SUM+BA:GOTO 200
315 A=INSTR(X$, "-") : IF A THEN 316 ELSE 200
316 PA=VAL(MID$(X$, A+2, 6)) : SUM=SUM-PA
317 IF (BA-PA) <=0! GOTO 200 ELSE PRINT#2, "***PAYMENT IN FULL IS REQUIRED EACH MO
NTH!***"
350 GOTO 200
1000 REM write total now due to invoice
1005 Y$=STRING$(48,32)+"Total Now Due: $"
1010 PRINT#2, Y$, SUM
1050 SUM=0!:LS$=""
1099 REM next bill
1100 GOTO 200
9000 CLOSE: PRINT TIME$
```

End Listing Five

$\rho R$   
8501  
BF  
1286 3413 5510 7672  
PR  
1345 3471 5569 7728  
TND  
1534 3658 5756 7913  
ALL  
769 838 894 2908 2977 5018 5074 7107 7176 7232 7301 7357 7426 7482  
CM  
25 25 25 35 30 50 65 45 45 45 45 65 65 45  
CSUMS  
75 65 115 355  
CBF  
290 50 -49 0  
CPR  
25 50 15 0  
CTND  
340 65 51 355

**Matrix A**

$\odot ((\rho BF), \rho CM) \rho CM$   
25 25 25 25  
25 25 25 25  
25 25 25 25  
35 35 35 35  
30 30 30 30  
50 50 50 50  
65 65 65 65  
45 45 45 45  
45 45 45 45  
45 45 45 45  
45 45 45 45  
65 65 65 65

**Matrix B**

$\text{ALL}^\circ. < \text{BF}$   
1 1 1 1  
1 1 1 1  
1 1 1 1  
0 1 1 1  
0 1 1 1  
0 0 1 1  
0 0 1 1  
0 0 0 1  
0 0 0 1  
0 0 0 1  
0 0 0 1  
0 0 0 1

**Matrix C**

$< \setminus (\text{ALL}^\circ. < \text{BF})$   
1 0 0 0  
1 0 0 0  
1 0 0 0  
0 1 0 0  
0 1 0 0  
0 0 1 0  
0 0 1 0  
0 0 0 1  
0 0 0 1  
0 0 0 1  
0 0 0 1

$+ f(< \setminus (\text{ALL}^\circ. < \text{BF})) \times \odot ((\rho BF), \rho CM) \rho CM$   
75 65 115 355

**Figure**

## ***Listing Six***

```

    ▽ PROCESS NAME;R;BF;PR;TND;ALL;A;CM;CSUMS;CBF;CPR;CTND;NP;MES
[1] DAI[2]
[2] NAME DNTIE "1 ◊ R<ONREAD "1 82 ,(ONSIZE "1),0 ◊ TIE FILES AND READ INTO WS
[3] BF<R FIND '+' ◊ PR<R FIND '-'
[4] TND<R FIND ': $'
[5] ALL<((R[A+1]="' ")^R[A-1]="' ")/A<R FINDALL 'igAW'
[6] CM<-, (CLEAN R[ALL..+ 28 29 30]), ' ' ◊ CLEAN REMOVES JUNK CONVERTS -+-
[7] CSUMS<+/(<\(ALL..(BF))X@((eBF),eCM)eCM ◊ SUM SESSIONS BY CLIENT
[8] CBF<-, (CLEAN R[BF..+1+13]), ' ' ◊ CPR<-, (CLEAN R[PFR..+1+13]), ' '
[9] CTND<CSUMS+CBF-CPR
[10] +(~V/NP<0)CPR-CBF)/L ◊ MES<NP/PR ◊ R[MES..+5+145]<***PAYMENT IN FULL IS REQUIRED EACH MONTH!***'
[11] L:R(TND..+2+14)<(2e(eTND))e'M(-)I4' OFMT CTND
[12] NAME[4+2x+/NAMEE':']<'0' ◊ NAME DNTIE "2 ◊ 0 ONRESIZE "2 ◊ R ONAPPEND "2
[13] ONUNTIE ONNUMS
[14] 'DONE!' ◊ DAI[2]
    ▽

    ▽ Z<R FIND F
[1] ASUBSTRING SEARCH OF R FOR F, GIVES LOCATION
[2] Z<(R OSS F)/1eR
    ▽

    ▽ Z<R FINDALL S
[1] AFINDS ALL OCCURRENCES OF S IN R
[2] Z<(RES)/1eR
    ▽

```

(Continued on next page)

# **GTEK**

DEVELOPMENT HARDWARE/SOFTWARE  
HIGH PERFORMANCE / COST RATIO  
INC. (601) 467-8048  
**EPROM PROGRAMMER**

Compatible w/all RS 232 serial interface port \* Auto select baud rate \* With or without handshaking \* Bidirectional Xon/Xoff and CTS/DTR supported \* Read pin compatible ROMS \* No personality modules \* Intel, Motorola, MCS86, Hex formats \* Split facility for 16 bit data paths \* Read, program, formatted list commands \* Interrupt driven, program and verify real time while sending data \* Program single byte, block, or whole EPROM \* Intelligent diagnostics discern bad and erasable EPROM \* Verify erasure and compare commands \* Busy light \* Complete w/Textool zero insertion force socket and integral 120 VAC power (240 VAC/50Hz available)

DR Utility Package allows communication with 7128, 7228, and 7956 programmers from the CP/M command line. Source Code is provided. PGX utility package allows the same thing, but will also allow you to specify a range of addresses to send to the programmer. Verify, set the Eeprom type.

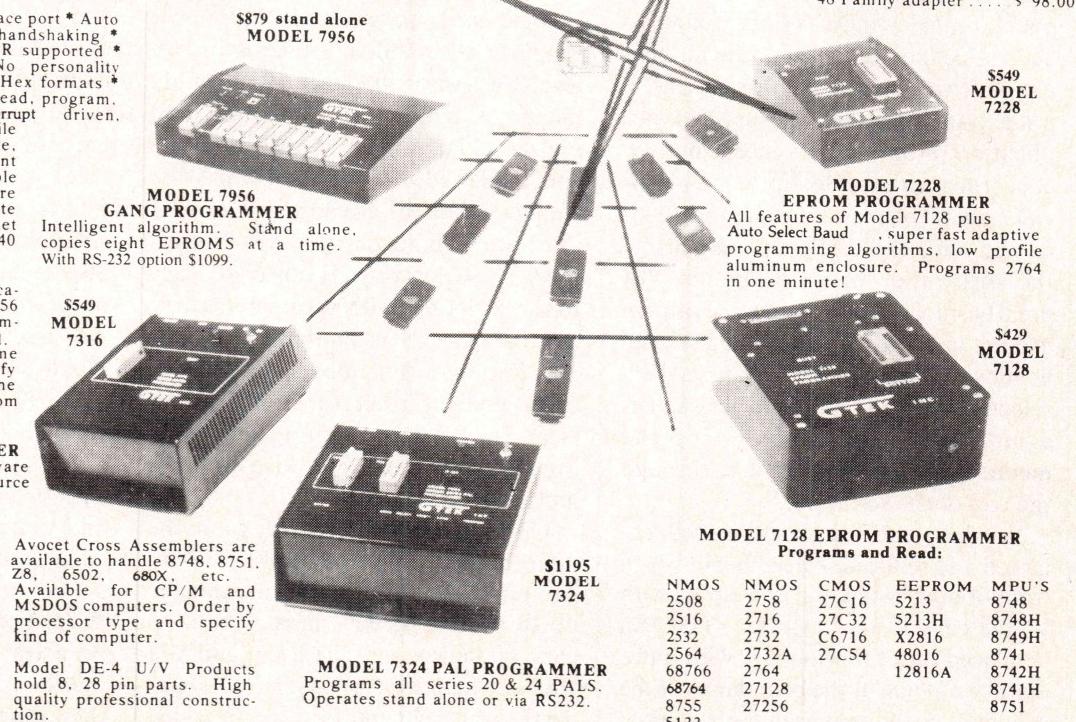
**MODEL 7316 PAL PROGRAMMER**  
Programs all series 20 PALS. Software included for compiling PAL source codes.

Software Available for CPM<sup>1</sup>, ISIS<sup>2</sup>

TRSDOS<sup>3</sup>, MSDOS.<sup>4</sup>

1. TM of Digital Research Corp.
2. TM of Intel Corp.
3. TM of Tandy Corp.
4. TM of Microsoft.

Post Office Box 289  
Waveland, Mississippi 39576  
(601)-467-8048



Circle no. 42 on reader service card.

## Listing Six

```
▽ Z←CLEAN V;Q
[1] ⍝ REMOVES ANY STRAY CHARACTERS AND CONVERTS NEG. TO MINUS
[2] Q←,V ⍝ Z←Q[(,~QE' 0123456789+-.')/~PQ]←' '
[3] Q←(P)V)⍝MINUS Q
[4] Q[(^/QE' ')/~1↑P;]←'0' ⍝ Z←Q
▽
```

```
▽ Z←MINUS M;N;RM;NM
[1] ⍝ CONVERTS "TO-" OR "-TO"; NO EFFECT IF "-" NOT PRESENT
[2] N←P RM←,M
[3] →(O(+/NM←,ME'-'')/L ⍝ →(O(+/NM←,ME'-'')/LL ⍝ Z←M ⍝ →O
[4] L:RM(NM/;N)←'-' ⍝ →E
[5] LL:RM(NM/;N)←'-' ⍝ DONT USE 'M(-)...' OFMT CAUSE
[6] E:Z←(P M)⍝RM ⍝ DONT KNOW FORMAT AHEAD OF TIME
▽
```

End Listings

**VSI – Virtual Screen Interface, Version 2.09**  
**Company: Amber Systems, Inc.,**  
**1171 S. Saratoga-Sunnyvale Rd.,**  
**San Jose, CA 95129**  
**Computer: IBM PC and direct compatibles**  
**Circle Reader Service No. 131**  
**Reviewed by Ronald G. Parsons**

VSI is a programmer's tool kit for building applications that manipulate the PC's screen to take advantage of overlapping windows and features such as color and borders. The authors of the program liken VSI for screens to a file system for disks. Using VSI, application programmers need not be concerned with the details of the physical screen, just as they are not concerned with the tracks and sectors of the disks they are writing on. VSI greatly facilitates the now common process of showing a menu on a pop-up window, prompting the user to make a selection with a pointing device such as a mouse or keyboard, erasing the menu, and prompting the user through the rest of the selection.

Windows can be created, moved, enlarged and reduced, placed behind or in front of other windows, and closed with simple calls to the interface. Text can be placed into the windows, with automatic wrapping at the boundaries of the window, at the programmer's option. Similarly, when the input cursor reaches the edge of the window, the

window will automatically scroll to keep the cursor in the window. The box around a window can be of any color, character, and attribute; this allows easy creation of "attention" windows for urgent messages. Windows are given a priority: windows of higher priority appear in front of those of lower priority. Best of all, the programmer does not need to bother with the details of all this screen action. Writing a program to do all this is very simple using VSI and produces a fast window action.

VSI supports both color and monochrome cards on the IBM PC, but not graphics. The VSI interface is in two levels: a level 0 interface to assembly language and a level 1 interface to high-level languages—Lattice C in the copy reviewed. Both interfaces provide similar functions because the level 1 interface is built on the level 0 interface. The C interface is especially easy to use: functions with parameters carry out the instructions on the screen.

Additional functions available include an exit reset function, several clear functions, a copy function, functions to read and write to a window, and many others.

Up to 255 different screens may be defined with sizes from one-by-one to 255 rows or columns. VSI can manage up to 65K of screen space. ANSI escape sequences may be used, and VSI will run under DOS 1.1 or 2.x. Diagnostic trace capabilities are provided for debugging. A VSI exerciser (VSIX) program that interprets commands

from a file can be used to demo the system or for help in understanding the VSI. Mice can be used as input devices to VSIX if they place cursor characters into the keyboard buffer.

### The VSI Distribution Kit

As indicated earlier, the version of VSI reviewed included an assembly language interface and a C interface configured for Lattice C V1.03. Other interfaces are available for Pascal (IBM or Microsoft), Compiled BASIC (IBM), Fortran (IBM), PL/I (DRI), and COBOL (Realia); these cost \$199 each. All necessary object modules are included to assemble/compile and link a program. In addition, source modules are supplied for the level 1 C interface, for assembly language routines to interface the C code, and for buffer assignment. The source code for VSIX is also included. This program contains many good hints for writing VSI code. An assembly language interface for making VSI a resident program allows the VSI interface to be accessed through a software interrupt.

A PC-size manual is provided, along with two reference cards for the level 0 and 1 interfaces. The documentation is clear and concise.

### A Test of Portability

As a test of the portability of the VSI interface, I converted the system to use the Computer Innovations (CI) C86 C V2.1 compiler rather than the Lattice C V2.0 compiler used in other parts of

this review. The C interface is supplied in a version tested with the Lattice C V1.03 compiler. Both the level 1 C interface code and the C-based exerciser are supplied in source form and were recompiled and linked with Lattice C V2.0 using the small model. No problems were encountered. Then the C interface code was compiled using the small model of CI C86. Since the only linkage between the C code and the rest of the interface is through one function, which passes the function parameters to the VSI system (this function is implemented in assembly language), the assembly language interface had to be reassembled. Because the assembly language interface is similar in Lattice C and CI C86, only the group and segment names in the supplied source needed to be changed to conform to their respective conventions. Once these two steps were done, my test programs were compiled and linked with CI C86, and they worked as before except for differences in the two C's I/O code. No changes were required in the VSI code. The steps were simple and took little time to complete. Thus, the VSI system appears to be quite portable among C compilers.

### Conclusion

If you will be writing a windowing application, seriously consider VSI. It is fast, easy to use, and full of capabilities, and it seems to be bug free.

### The Fancy Font System, Version 2.0

**Company:** SoftCraft, Inc., 222  
State Street, Madison, WI  
53703 (606) 257-3300

**Price:** \$180

**Computer:** CP/M, Apple CP/M, IBM  
PC, Osborne, Kaypro, Victor  
900, Epson QX-10 computers  
and Epson MX, FX, RX, Gemini  
10X, IBM Graphics Printer,  
Riteman Inforunner, TI 855/  
850, or C. Itoh and NEC8023  
(IBM only) printers

**Circle Reader Service No. 133**

**Reviewed by David D. Clark**

The Fancy Font System is a personal typesetting system for creating nicely printed documents using a personal

computer and a dot matrix printer. It uses the high-resolution graphics capabilities of popular printers to produce high quality printing. You can mix different font styles, sizes, and special symbols in the same document. You can use the fonts supplied or create and customize your own alphabets and special symbols.

The Fancy Font System can no longer be considered a new product; you have surely seen the advertisements by now. I've had a copy for over a year. I originally bought version 1.7 of the system, and frankly it was a pain to use. It did produce beautiful print but only after a lot of work. The purpose of this article is to review the updated system. The new version 2.0 is a real improvement. It is much easier to use, has lots of additional options and controls, includes improved documentation, and is available on more computers and printers. If you have an Epson FX-80 printer, the news is even better: it produces even higher quality print.

### System Contents

What you get depends on the computer and printer you use. I have a CP/M 2.2-based system with an Epson FX-80 printer. My update from version 1.7 consisted of a new user manual, a quick reference card, and two disks containing the font printing, editing, and creating programs as well as a number of fonts and the Hershey character data base. Also included were files to automatically create some additional fonts that would not fit on the disks.

Besides the standard fonts supplied with the system, SoftCraft is building a font library from which it will sell additional fonts. They are actively soliciting users to submit any fonts that they may have developed themselves. The company also puts out a periodic newsletter informing users of new developments and tips on using the system.

### Documentation

The user manual and quick reference card are excellent examples of some of the capabilities of the system: they were printed with Fancy Font. They are very pleasant to the eye, besides containing well-written documentation for the system.

The manual starts with an introduction followed by three large sections,

# C

## Software Development PCDOS/MSDOS

### Complete C Compiler

- Full C per K&R
- Inline 8087 or Assembler Floating Point, Auto Select of 8087
- Full 1Mb Addressing for Code or Data
- Transcendental Functions
- ROMable Code
- Register Variables
- Supports Inline Assembler Code

### MSDOS 1.1/2.0

### Library Support

- All functions from K&R
- All DOS 2.0 Functions
- Auto Select of 1.1 or 2.0
- Program Chaining Using Exec
- Environment Available to Main

### c-window™

### Symbolic Debugger

- Source Code Display
- Variable Display & Alteration Using C Expressions
- Automatic Commands
- Multiple Breakpoints by Function & Line Number

### 8088/8086 Assembler

- FAST — Up to 4 times Faster than IBM Assembler
- Standard Intel Mnemonics
- Compatible with MSDOS Linker
- Supports Full Memory Model

### 8088 Software Development Package

**\$199<sup>00</sup>**

Includes: C Compiler/Library,  
c-window, and Assembler, plus  
Source Code for c-systems Print  
Utility

**c-systems**

P.O. Box 3253  
Fullerton, CA 92634  
714-637-5362

Circle no. 16 on reader service card.

one for each of the three programs, and a number of appendices. The documentation follows the same format for each program. Each section starts with an introduction outlining the purpose and use of the program. After the generalities are taken care of, the manual describes the commands in detail, devoting one or more pages to each. The detailed descriptions consist of usage instructions, examples, and notes. Depending on the command, there may also be sections on default values, legal values, errors, and suggestions.

The appendices contain a glossary, font descriptions and samples, hints for using Fancy Font with word processors, a summary of error messages, a description of the distribution files, a print-out of the Hershey data base, a description of data file formats, the ASCII character set, and a parameter and command summary. The quick reference card is keyed to the manual by page number; if the card doesn't give you all the information you want, it is simple to look it up.

### Pfont

Pfont, the printing program, is the program you will probably use most often. Before using Pfont, you must use your word processor or text editor to create a file containing embedded Pfont formatting commands. Then you start Pfont and set up the printing parameters.

There are 27 embedded formatting commands, usually with several variations available for each one. They control things like font selection, horizontal and vertical movement, underlining, justification, word wrap, indentation, paging, centering, and so on. In general, you use the formatting commands the same way you would in a powerful word processor. Pfont is more advanced than a typical word processor in some ways but more primitive in others. For example, how many word processors can you tell to switch from an 8-point font to a 40-point font or to move backwards up the page 2.63 cm before printing the next line? On the other hand, it's pretty tough to create a five-line page header with Pfont.

Once you have prepared the text file containing the formatting commands, Pfont is executed to print the file. Before printing begins, you have the opportunity to enter a number of printing

parameters. About 30 printing parameters control things like print quality (and speed), number of copies, line width, margins, first and last page to be printed, pause after each page, page length, and so on. You are required to enter only the names of the files to be printed and the names of the fonts to be used during printing; all of the other parameters have reasonable default values. The parameters can be entered interactively or retrieved from a parameter file.

Because of the relatively heavy use Pfont will receive relative to the other two programs, it has been constructed with a powerful user interface. You can obtain help with individual parameters and review the settings of all parameters.

Once printing has started, another group of commands becomes active. These commands will immediately stop printing, skip to the next page, skip to the next file, pause after the current page, or allow the user to change the draft mode.

### Efont

Efont is the font editing program. It can be used to alter existing fonts or to create new ones from scratch. Of the three programs, this one is probably the most difficult to use—not because of inherent problems in the program but because the mechanics involved in editing a font can become quite involved.

The fundamentals of the process involve loading a font, creating expanded versions of some characters in a group of ASCII text files, editing those files, replacing elements of a font with the edited version, and saving the altered font. It is also possible to create characters from scratch by creating an ASCII text file of the proper type and using the "replace" command for an otherwise empty font file. Additional commands print characters, alter font information, move a character set up or down (to create super- and subscripts, for example), modify the margins for a set of characters, and so on.

When a particular character or range of characters is to be edited, the program will create a series of ASCII text files, one for each character, that contain a line of information about the character (the left and right margins and a value related to the height of the

character) followed by the actual character represented as asterisks. One asterisk in the text file represents one pin strike in the finished character. To alter a character, you must exit the Efont program and use a text editor to add, delete, or move asterisks around and to make appropriate changes to the information line. When the changes are complete, the Efont program is entered again to read the altered text file and create an internal character representation from it.

It is sometimes difficult to get a good feel for how the finished character will look: the aspect ratio of the text file is different than the aspect ratio in the printed character. By aspect ratio I mean the ratio of horizontal to vertical elements in a unit square. For example, when Pfont is used to print a character on my printer, there are up to 240 horizontal positions by 216 vertical positions per square inch;  $240/216$  yields an aspect ratio of about 1.11. However, on my terminal that same square inch can have about nine horizontal elements and four vertical elements, giving an aspect ratio of 2.25. Because of this difference in aspect ratios, the character in its text file representation always looks taller and thinner than it will look when printed.

In practice, it is not difficult to make simple changes to only a few characters. An example in the user manual runs through the process of adding a tilde to an *n*. It is a fairly simple process and well explained. As your editing needs become more complex, it will be necessary to learn some technical aspects of the data representations and conventions used for character fonts. Most users, however, will probably never need to use this program.

### Cfont

Cfont is used in conjunction with the Hershey character data base to create new fonts and special symbols. The Hershey data base was created by Alan V. Hershey for the National Bureau of Standards. Besides various styles and sizes of Roman alphabets, it contains old English styles, Greek alphabets of different sizes, and even the Cyrillic (Russian) alphabet. There are also a number of special symbols and graphics from mathematics, music, and who knows where else.

Creating a font from the Hershey data base is well described in the manual and really very easy. It consists of creating a mapping between the number of a character in the data base and a corresponding ASCII number. Then you supply Cfont with scale factors that determine the size of the font created and a baseline—almost always a seven for technical reasons explained in the manual. The process can be performed interactively, or you can create a file of mappings. Examples of both methods are provided. My disk came with map files and a submit file to automatically create three sizes each of a script font and an old English font.

### General Impressions

With any complicated program, you can usually expect some problems. I found one, but I'm not sure about it. When I updated my system from version 1.7 to 2.0, a font conversion utility was provided to create fonts from my old fonts that would be compatible with the new version. The copy of the program I got didn't work correctly when parameters were supplied from the command line. I bashed a couple of files this way before switching to the interactive mode. Then things worked fine. I don't know if the problem was with the program or just with my copy, but since it worked correctly in the interactive mode, I didn't bother to get a new copy from SoftCraft.

The biggest drawback to Fancy Font is speed. Although it prints beautifully, it prints slowly. This isn't really a problem with the program, though; it's a limitation of the printer. The program uses the graphics capabilities of the printer, driving it as fast as the printer can run. In the highest quality print mode on an FX-80 printer, the program makes at least six passes over a line of text, usually more, depending on the size of the letters. It can take quite a while to print out a page. Versions configured to the Toshiba 1350 or Epson LQ-1500 can print in one pass according to the manual and, I presume, would be faster. (Although the Toshiba and LQ-1500 printers are not mentioned in the advertising, they are discussed in the user manual. You should probably call SoftCraft to find out if versions for those printers are available.)

Another difficulty on my system is the size of the font files. When Pfont prints a file, it loads as many fonts as it can into main memory and reads pieces of the rest from disk as it needs them. My 64K CP/M system usually has room to keep only one font in memory. My disks, however, are fast enough to just about keep up with the printer when Pfont is reading font information from the disk. On systems with slower disks, I have seen the print speed decrease dramatically. For printing rough drafts, the selection can be changed so that all of the information in a font file need not be read in. On IBM systems with larger memories or on systems that use a printer with lower resolution than the FX-80, this may not be a problem.

This program produces very nice print, but in general it is not quite as good as that produced by a daisy-wheel printer. On my printer, the horizontal resolution is 240 dpi (dots per inch) with a vertical resolution of 216 dpi. Although human eyes have a resolution of over 1000 dpi under favorable circumstances, ink on paper usually cannot be resolved that finely. On relatively rough paper, such as that used in computer printers as opposed to that used to print *Scientific American*, the highest resolution appears to be about 400–500 dpi. Fancy Font approaches this resolution but can't quite achieve it. This is most noticeable on lightweight diagonal strokes. On vertical and horizontal strokes, the printed image is perfect, as you might expect; the individual pin strikes cannot be discerned. One method to overcome the resolution limits of the printer is to print the text larger than desired then reduce the image. The user manual, which was created this way, looks good. The justified, proportionally spaced output of the program looks better to my eye than that produced by most word processors with daisy-wheel printers. Using Fancy Font is also considerably more fun.

In summary, the program produces good output and works reliably. Except for the glitch with the font conversion utility, I have never had a problem with the system. The new additions made to version 2.0 make the system considerably more flexible and easy to use.

DDJ

### CUCUMBER INFORMATION SYSTEMS

5611 Kraft Drive  
Rockville, MD 20852  
(301) 984-3539 (301) 881-2722



### A BREAKTHROUGH IN INFORMATION RETRIEVAL

You no longer have to know exactly what you want

to find what you need —

#### Retrieve information by:

- Natural Language
- Heuristic Word Associations
- Word Roots
- Like Documents
- Full Boolean Logic and Adjacency
- Truncation and Wild Cards
- Specific Fields

#### And don't overlook these features:

- Up to 256 Fields Per Document
- No Limits on Document or Field Lengths
- Automatic Full-Text Indexing
- Sort Documents on Any Field
- All Fields Variable Length
- Full-Screen Data Entry

Add SIRE's "ranked output" - documents ordered by their probable usefulness - and the result is the best retrieval system at ANY price on ANY computer.

SIRE runs on the IBM PC, XT and AT, DEC PDP-11 and VAX, and other 16 and 32 bit computers running UNIX, MS-DOS and RSX.

Order the SIRE Demonstration System and Manual (for IBM PC/XT) for \$25

For further details contact:

Dave Harris  
Cucumber Information Systems  
5611 Kraft Drive  
Rockville, Maryland 20852  
(301) 984-3539 (301) 881-2722

SIRE, MS-DOS, UNIX and VAX are trademarks of KNM, Inc. Microsoft, Bell Laboratories, and Digital Equipment Corporation, respectively.

\*\*\*\*\*

### CUCUMBER BOOKSHOP, INC.

5611 KRAFT DRIVE  
ROCKVILLE, MARYLAND 20852

The leading Unix &  
C Language Bookstore  
in the Nation

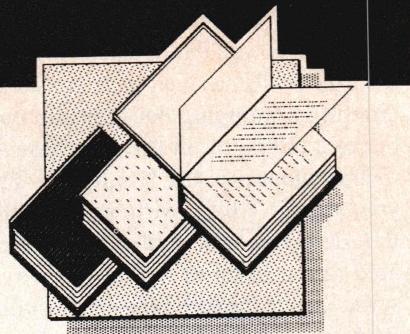
Mail your order today  
or call  
Cucumber Bookshop  
(301) 881-2722



FOR  
**Your New  
UNIX\* & C  
Book List**

\*UNIX is a Trademark of AT&T Bell Labs.

Circle no. 21 on reader service card.



## The C Programming Tutor

by Leon A. Wortman and

Thomas O. Sidebottom

Published by Robert J. Brady Co.

274 pages, paperback

Reviewed by Ian Ashdown

This is most decidedly not your average book on the C programming language. While the authors follow the usual format of explaining the features of the language with examples, it is the examples that set this book apart from all others.

Kernighan & Ritchie in *The C Programming Language* (Prentice-Hall) may have set the stage for *The C Programming Tutor* through their own examples. These included a fully functional word frequency utility using binary trees and a very elegant demonstration of recursion. Wortman and Sidebottom, however, have taken the concept of instructive examples one giant step further by presenting nontrivial and useful programs to demonstrate C. They explain each feature of the language in the context of an overall program rather than as an isolated example.

To be more specific, these programs include (in order of appearance and complexity) a simple calculator, a histogram generation utility, a file encryption program, a printer configuration utility, three text readability analyzers, a word frequency analyzer, a C program cross-reference listing utility, and a chart generator for C program function calls. By the time the authors get around to presenting the function call chart generator, they have interspersed the text with some very complex C code. It is all explained in detail, however, so that you can understand and learn from the listings as you key them into your computer.

The book comprises two parts: "Tutorial" and "Useful Programs." The tu-

torial takes you by the hand through the usual programming examples to demonstrate the various features of the language—but with a firm emphasis on solving problems. Each example is stated first as a problem with the solution blocked out in English-language statements. The following text then shows how you can implement these statements in C. Very much in the spirit of C, the authors use the resultant functions later for the solution of more complex problems. As more interesting features of C are discussed, many of the functions are progressively rewritten to include them.

I should say here and now that I do not recommend *The C Programming Tutor* as a first and only guide to learning the C language. Learning how to program in C is not particularly easy, and the fast pace of this book is likely to leave the novice confused and frustrated. Learn C from a book like Jack Purdum's *C Programming Guide* (Que Corp.) then use this book to polish your skills.

Since this book teaches C by way of programming examples, I also do not recommend it as a reference guide to the language. While most of C is described in detail, the information is subordinate to the programs. Unless you read the book from cover to cover, you will have a hard time trying to find specific information about some of C's features. As an example, looking up "#define" tells you only that it is a compiler directive used for defining constants. You have to know already about its macro capabilities to realize that the index entry for "code macros — expanding the macro" is an important related topic.

One nice touch is the ongoing discussion of various C compilers, including both Unix and microcomputer implementations. While C is nominally "the" transportable compiler, every

compiler writer seems to have his or her own ideas about compatibility. The authors frequently warn you that your particular compiler, especially microcomputer versions, may not support the feature currently under discussion.

The chapter on strings, whimsically named "Things Called Strings," presents a bonus: complete implementations of strlen, strcpy, strcat, strcmp, and index. A recent trend among the companies distributing C compilers is to sell the source code to the standard library functions at additional cost, if at all. This is most unfortunate, for this code is an invaluable resource when it comes to learning the language and writing variations of these functions. While there are many more functions in the standard library, the code presented is nonetheless instructive.

"Things Called Strings" also presents the source code for a pattern-matching utility that finds a string in an ASCII file. While having nowhere near the power and flexibility of the Unix command grep, it is again instructive as a program example. With a bit of programming initiative, you could expand it into something quite useful.

The next chapter, entitled "Paths And Pointers," illustrates the interesting approach the authors have taken to this often confusing topic. *The C Programming Tutor* likens pointers to paths that lead to various places. Between this analogy and the illustrations, the mystery of pointers is clearly explained.

The tutorial part of the book ends with the histogram generation utility, version 10. Part Two, "Useful Programs," is a bit of a misnomer, for it covers such essential C language features as the preprocessor's macro capabilities (the preprocessor itself is never discussed), memory allocation, structures (in two pages!), initialization, typedefs, and buffering. Other features such as separate compilation

of modules, storage classes, pointers to functions, external functions, unions, and bit fields are ignored altogether.

If you know anything at all about C, you may think from the last statement that *The C Programming Tutor* is somewhat lacking as a tutorial on the language. The authors fully admit this—they did not intend for their book to cover every aspect of C. The purpose of *The C Programming Tutor* is to teach the reader how to program effectively in C, not to teach the language *per se*.

Five major programs are presented, ranging from EPSET, a simple utility program for configuring software-programmable dot matrix printers, to CALLS, which prints a chart of how functions in a C program call one another. (To be candid, CALLS is the reason I bought this book. It has proven useful as a debugging and documentation tool, well worth the price of the book in itself.)

Three versions of a text analysis program are given. One is an extended version of the Unix command wc (word count) that counts the number of characters, words, lines, and sentences in a file. The second version applies a definitive "readability" formula from Rudolf Flesch's *The Art of Readable Writing* (Harper & Row) that uses the count of words, average sentence length, syllable count, names, and personal pronouns to determine the ease of reading and human interest content of a body of text.

The third version uses the same formula but optimizes its performance by substituting a hash search method for the linear method used in the second version. The discussion on hashing is very instructive and produces some interesting code that can be adapted for use in many other programs.

The next program is the word frequency analyzer WFREQ. While of some interest, this program is eclipsed by Kernighan & Ritchie's version in *The C Programming Language*. The inherent sorting of words performed by their recursive transversal of a binary tree is far more interesting than the hash table entry and shell sort methods used here. (With the power of recursion available in C, Wortman and Sidebottom did not bother to discuss Quicksort . . . Shame!)

XREF is a C program cross-reference listing utility that reads a file of C source code, adds line numbers, and makes an alphabetically sorted list of all the names (tokens) used in the file, along with a record of the line numbers associated with each name. Apart from its obvious usefulness in debugging and documenting other C programs, XREF affords the authors a chance to discuss such topics as linked lists and compound data structures. The code itself is quite complex but well documented. As with the text analysis programs, some of the developed functions are useful in their own right.

Finally, there is CALLS. As stated earlier, this program is the reason I bought *The C Programming Tutor*. In operation, CALLS scans a C source code file looking for functions. When it finds one, it scans the body of the function for calls made to other functions. When it has done this for all functions, a chart is printed showing the functions in their order of occurrence. Function calls are indicated by indenting the names, recursive functions are identified, and repetitively called functions have their first occurrences referenced by line number. A key advantage of the call chart output is that it is a graphic display of the program structure.

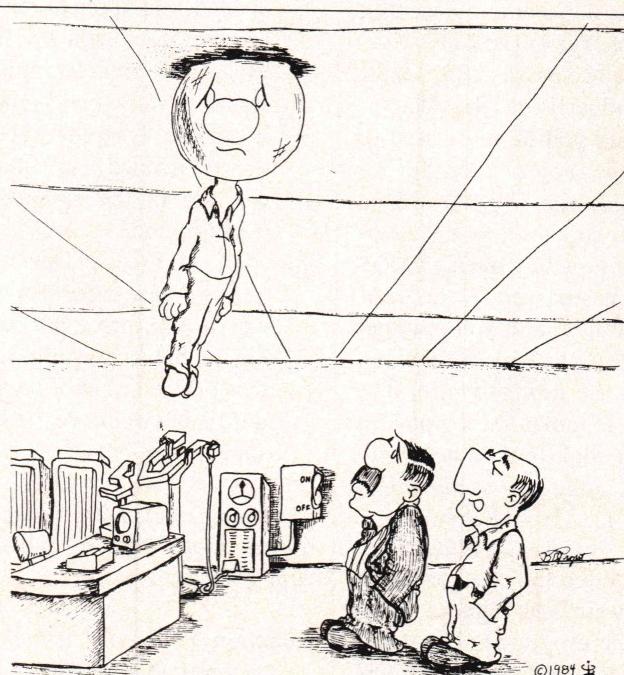
The book finishes with four appendices. Appendix A discusses the various

C compilers available (as of July 1983) for CP/M-80, CP/M-86, and MSDOS. The information presented is interesting but was already out of date by the time the book was published. Only one thing remains constant in the world of microcomputer C compilers: they will all undergo continual improvement until they are fully compatible with their Unix brethren.

The remaining three appendices discuss the C header files ctype.h and math.h. The source code is given for ctype.h, which defines the character classification and conversion functions (isascii, iscntrl, toascii, etc.) of C's standard library. The source code is also given for math.h, but the transcendental functions are declared as "external." The authors simply note that if a compiler supports float and double data types but does not provide mathematical functions, an experienced programmer can write them in C and declare them using math.h.

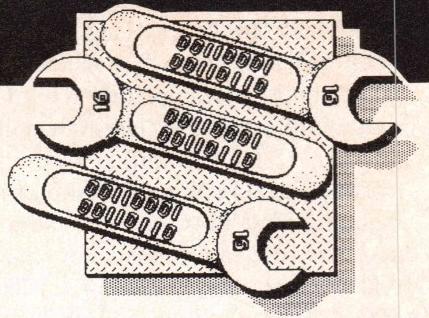
Conclusion? If you program in C, buy this book for CALLS and XREF alone. The rest of the book will be of varying usefulness and interest to different people, but most will learn and profit from reading it.

DDJ



HE WAS WORKING ON A NEW CONCEPT IN BUBBLE MEMORY WHEN THINGS GOT OUT OF HAND.

# 16-BIT SOFTWARE TOOLBOX



by Ray Duncan

## PCDOS version 3.0

Along with the recently announced PC/AT computer, IBM announced release 3.0 of PCDOS. Together with a bad case of bloat (version 3.0 occupies some 48 kilobytes of memory on a PC/AT with hard disk), the new release of PCDOS included a number of additional or modified function calls compared to version 2. Since the DOS Technical Manual is still a very scarce commodity, I will provide an overview of the significant changes in this month's column.

One of the most interesting features of the release of PCDOS 3.0 is the virtual disappearance of its originator, Microsoft, into the background noise. In fact, the word 'Microsoft' doesn't appear at all in sign-on messages or the various DOS manuals, and can be found only in very small print on the diskette label. Although IBM never discusses its future product plans, still, if you were in charge of the Entry Systems Division and you saw Bill Gates on television advertising the Macintosh, what would you be motivated to do? More on this subject later, but reflect on the fact that TopView is a high-performance, windowing environment that completely hides PCDOS and has been earmarked as an IBM "Strategic Product," and you may get an idea which way the wind is blowing.

But back to the new software. The Critical Error Handler interrupt (Int 24H) has been slightly modified, with the addition of some new status information passed in AH and a new DOS response upon return (the application can ask for the offending system call to be "failed" as well as just ignore, retry, or terminate the program). The documentation on how to field the critical error interrupt, which was rather sketchy before, is still somewhat cryptic, but has been expanded to the point

of being comprehensible.

Interrupt 2FH is new, and is essentially an interface between application software and the print spooler. A user program can add files to the print queue, cancel files, or examine the spooler's list of waiting files.

Under the general heading of Interrupt 21H (DOS function calls), the following three functions were modified:

Function 38H (Get or Set Country Information) was radically expanded: first, with the ability to set a country code as well as to interrogate it; next, with the capacity for more than 255 country codes; and last, with much more information passed back in the data block regarding country specific delimiters and formats.

Function 3DH (Open File) was enhanced to support both multi-tasking and networking. When a file is opened, the application can specify whether it will be "inherited" by a child process running in the same network node, and whether it can be "shared" (i.e. independently opened) by another task running in the same or another network node.

Function 44H (I/O Device Control) has two new capabilities: to interrogate whether a particular block device has removable media, and to set the number of retries and the delay between retries for "file sharing conflicts."

There are also five completely new documented function calls for Interrupt 21H:

Function 59H (Get Extended Error) can be called by an application after an error code is returned by some other function to get more detailed information about the

failure, the class of failure (temporary, system internal, hardware, etc.), and the recommended response for the application (retry, delay and retry, abort, etc.)

Function 5AH (Create Temporary File) is passed a path string by the application, generates a unique filename string, creates the file in the specified subdirectory, and returns a complete path and file specification string. The file can then be opened by the application and used in any way desired. The file is not deleted automatically when the application terminates.

Function 5BH (Create File) is exactly like the DOS 2.0 function 3CH (Create File), except that it will fail if the file already exists. Function 3CH, on the other hand, truncates a previously existing file by the same name to zero length, then returns a success code.

Function 5CH (Lock/Unlock File Access) provides a general mechanism for a task to gain exclusive access to a region of a file, even if the file is being shared with other tasks.

Function 62H (Get Program Segment Prefix Address) is self-explanatory. This will be handy for writers of EXE-type programs.

With release 3.0 of PCDOS, it is indeed clear that this operating system is going to continue to get more and more complex and powerful, even if it (or its descendent) is eventually completely submerged in TopView or some other IBM-proprietary shell. I suggest that the following guidelines for "well-behaved" applications will make the programmer's life easier in the brave new world of multi-tasking and networking that is nigh upon us:

- Use the Extended File functions

(3CH-42H) to open, close, read, and write files, rather than the old "FCB" type calls which were cloned from CP/M. The extended functions are more powerful, have better error reporting, and support the hierarchical file structure and file sharing.

- Use the Modify Allocated Memory Blocks call to release any memory that is not required by the application program. In the same vein, be sure to examine the program segment prefix to find the amount of memory allocated to your application, and do not access memory outside those bounds (even if you can establish that more memory is physically present).

- Don't access the interrupt vectors directly; use the DOS function calls 25H and 35H to set and get the contents of interrupt vectors respectively. This has implications for TopView as well as for the 80286 in the PC/AT, which can support multiple tables of interrupt vectors in protected mode.

- When your program gets control, use function 30H to get the DOS function number. If your program is running under DOS 3.x, interrogate the extended error codes (function 59H) when a DOS function fails to get more detailed information and the suggested action. In any case, employ the critical error handler interrupt capabilities to make your program more forgiving of hardware errors.

- Use the Exec call to spawn other tasks or load overlays; don't try to set up Program Segment Prefixes yourself.

- Write directly to the video buffer if you must, but perform all mode changes through calls to the ROM BIOS. Cause all video code to access a variable that contains the buffer segment address. This will make modification of a program for operation under TopView (where a "shadow buffer" is assigned) much simpler.

- If writing to a standard device or a file, send strings rather than single characters whenever possible. Take advantage of the buffering and fast transfer logic written into the DOS drivers. Conversely, don't waste your program's resources on internal blocking and deblocking of records or characters when the DOS is already imposing multiple layers of buffering between you and the device.

- Before exiting the application, be

careful to close all file control blocks and/or handles. If regions of a file have been locked, unlock them before closing the file.

- Terminate your program with a return code via function 4CH of Interrupt 21H, rather than the previously accepted mechanism of function 0 or Int 20H. The return code can be interrogated by the invoking process or by the batch subcommands IF and ERRORLEVEL.
- If writing a resident driver, use Interrupt 21H Function 31H to terminate and stay resident, rather than the pre-

viously approved Interrupt 27H. This allows for return information to be passed, and also for resident programs larger than 64 kilobytes.

## Graphics Routines for the IBM PC

Bruce A. Smith writes: "Enclosed is an 8088 assembly listing of a high performance line drawing and point plotting routine for the IBM PC or PCjr. These routines were written for utilization in application programs that I am devel-

# WHY WOULD ANY SANE PERSON SPEND \$199 FOR A BetterBASIC SYSTEM WHEN DOS's IS FREE?

HERE ARE 10 REASONS:  
TEST YOUR SANITY

1. Full support for 640K memory
2. Structured language with BASIC syntax
3. Separately compiled program modules
4. Speed: FAST
5. Extensibility (Make your own BASIC.)
6. User-defined procedures and functions
7. Built-in windows support
8. Interactive programming language based on an incremental compiler
9. 8087 math support
10. Runs on IBM PC, IBM PC/XT and compatibles

Summit Software Technology, Inc.<sup>TM</sup>  
P.O. Box 99  
Babson Park  
Wellesley, MA 02157  
**(617) 235-0729**

BetterBASIC is a trademark of Summit Software Technology, Inc. IBM PC, IBM PC/XT and PC/DOS are trademarks of International Business Machines Corp. MS-DOS is a trademark of Microsoft Corp.

NOW AVAILABLE FOR  
THE TANDY 2000 & 1200

**Better**  
**BASIC**<sup>TM</sup>

Sane Programmers  
Order BetterBASIC Now

Price: \$199  
8087 Math Module: \$99  
Runtime System: \$250  
Sample Disk: \$10

MasterCard, VISA, P.O. Checks,  
Money Orders, and  
C.O.D. accepted.

Circle no. 98 on reader service card.

oping in 'C'. I wish to encourage the integration of graphics in software and I thought these two fast 'nuts and bolts' plotting routines were worth sharing with your readers.

"Listing One (page 110) contains the two procedures to plot a point or draw a line on the medium resolution screen of the PC. The line drawing routine uses self-modifying code. They are both written to OR the color bits onto the screen. The line drawing routine is approximately three times faster than the one listed in Morgan's *Bluebook of Assembly Routines for the IBM PC & XT*. It is also three times faster than the line drawing routine used in the ROM Basic on the PCjr. The point plotting routine is a little more difficult to compare, because the overhead of the call and the computation for the next point become significant. In performance tests, it plotted points half again as fast

as the routine in Morgan's book, and in spite of the overhead, it performed four times faster than the ROM BIOS point plotting routines.

"Listing Two (page 122) is a 'C' program which illustrates use of these routines from a high level language." Interested readers can contact Bruce Smith at 305 E. Edgewood Blvd., Apt. 5, Lansing, MI 48910.

The line drawing routine sent by Mr. Smith employs Bresenham's Algorithm, which is particularly well suited to microcomputer graphics routines because it has a simple inner loop and uses only integer arithmetic. It was first published by J. E. Bresenham in the article "Algorithm for Computer Control of Digital Plotters," *IBM Systems Journal*, 4 (1) 1965, pages 25 - 30.

Those of you delving into graphics on any microcomputer will find the fol-

lowing two books to be invaluable:

*Fundamentals of Interactive Computer Graphics*, by J. D. Foley and A. Van Dam, Addison-Wesley, 1982. This book is the standard by which all future graphics texts will be measured. Incidentally, it contains a nice explanation of the Bresenham algorithm on pages 433 - 436. on pages 433 - 436.

*Applied Concepts in Microcomputer Graphics*, by Bruce A. Artwick (author of the IBM PC Flight Simulator program), Prentice Hall, 1984. No-nonsense, practical advice from a guy who has done it all.

DDJ

**Reader Ballot**

Vote for your favorite feature/article.  
Circle Reader Service No. 196.

## 16-Bit (Text begins on page 108)

### Listing One

PAGE 255,132

```
;;
; Plot Point or Draw Line on IBM Color Graphics Adaptor
;
; "orline.asm"
; "orpt.asm"
;
; written by Bruce A. Smith 7/24/84
;
;
; DGROUP GROUP DATA
DATA SEGMENT WORD PUBLIC 'DATA'
ASSUME DS:DGROUP
PUBLIC COLOR,X1,X2,Y1,Y2
;
; the order and type of declaration is important here
;
y2 dw 0
x2 dw 0
y1 dw 0
x1 dw 0
color db 0
;
; color = color * 4 + 1, color 0 == mask
;
ctableh db 03Fh,0CFh,0F3h,0FCh
db 040h,010h,004h,001h
db 080h,020h,008h,002h
db 0C0h,030h,00Ch,003h
;
fakedw = 1000h
;
DATA ENDS
;
```

(Continued on page 112)

# ARTIFICIAL INTELLIGENCE Programming:

Learn Fast,  
Experiment,

with the language chosen by JAPAN.

## PROLOG-86™

is "Standard" with Tutorials, samples of

- an Expert System
- a Natural Language

and other sample programs

1 or 2 pages of Prolog would take 10+ pages in C or PASCAL - a different way of thinking. Become familiar in one evening.

**CONTEST: \$1,000 award.**

New deadline 4/30/85.

Intro Price: \$125 for CPM-86, MSDOS.  
Plus \$3 Shipping.

Full Refund if  
not satisfied in  
first 30 days.

**617-659-1571**

**Solution  
Systems**  
335-D Washington Street  
Norwell, MA 02061

Circle no. 94 on reader service card.

## C **Helper**™

FIRST-AID FOR C  
PROGRAMS

Save time and frustration when  
analyzing and manipulating  
C programs. Use C HELPER's  
UNIX-like utilities which include:

DIFF and CMP - for "intelligent" file  
comparisons.

XREF - cross references variables by  
function and line.

C Flow Chart - shows what functions  
call each other.

C Beautifier - make source more regular  
and readable.

GREP - search for sophisticated pat-  
terns in text.

There are several other utilities that help  
with converting from one C compiler to  
another and with printing programs.

C Helper is written in portable C and in-  
cludes both full source code  
and executable files for  
\$135 for MS-DOS, CPM-80  
or CPM-86.

Use VISA, Master Card  
or COD.

Call: **617-659-1571**

**Solution  
Systems**  
335 Washington Street  
Norwell, MA 02061

Circle no. 95 on reader service card.

**BRIEF**

## Program Editing is finally both Intuitive and Powerful

... and configurable to suit your style

BRIEF lets you concentrate on programming by keeping the Editor  
"out of the way," while combining power and natural flow:

**The New Standard.** No longer does an Editor have to be "in your way" to provide full power. By combining power with natural flow, the new advanced BRIEF is in a class by itself.

**BRIEF lets you concentrate on programming.** Your thoughts flow smoothly, intuitively. 15 minutes is all you need to become fully productive. You can then do precisely what you want quickly, with minimum effort and without dull repetitions.

**BRIEF adapts to your style.** You can use BRIEF without modification, because it's distributed with an "ideal" configuration. Or you can make any change you want, add any feature of your own. Reconfigure the whole keyboard or just the Function Keys. Change the way the commands work or just the start-up defaults.

**Availability:** PCDOS-compatible systems with at least 192K and one floppy drive are required. Though your initial copy is protected, an unprotected version is available when you register BRIEF. Ask for special IBM AT or Tandy 2000 versions.

**Pricing:** Only \$195 . . . with discounts for volume end-users. A demonstration version is available for only \$10 and can be available towards any Solution Systems purchase.

**Win \$1,000** and substantial recognition for the Outstanding Practical BRIEF Macro. Other awards will also be given. Macros are fully programmable.

---

### BRIEF'S PERFORMANCE IS NOT EQUALLED IN MICROS, MINIS AND MAINFRAMES

- Full UNDO (N Times)
- Edit Multiple Large Files
- True Automatic Indent for C
- Exit to DOS Inside BRIEF
- Uses All Available Memory
- Intuitive Commands
- Tutorial
- Repeat Keystroke Sequences
- Windows (Tiled and "Pop Up")
- Unlimited File Size
- Reconfigurable Keyboard
- Online Help
- Search for Complex Patterns
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery

**PLUS a Complete, Powerful, Readable, Compiled MACRO Language**

---

**RISK-FREE Editing** is possible because of UNDO. All operations can be undone, including global revisions, massive deletions, typing, etc. Up to 100 such operations can be undone. Experiment while you edit. No other known micro software product has such a powerful UNDO.

Try BRIEF. Use the Demo . . .  
or the full product for 30 days.

Call or write us . . .  
**617-659-1571**

BRIEF is a trademark of UnderWare.

Solution Systems is a trademark of Solution Systems.

**Solution  
Systems**™

335-D Washington St., Norwell, MA 02061

Circle no. 75 on reader service card.

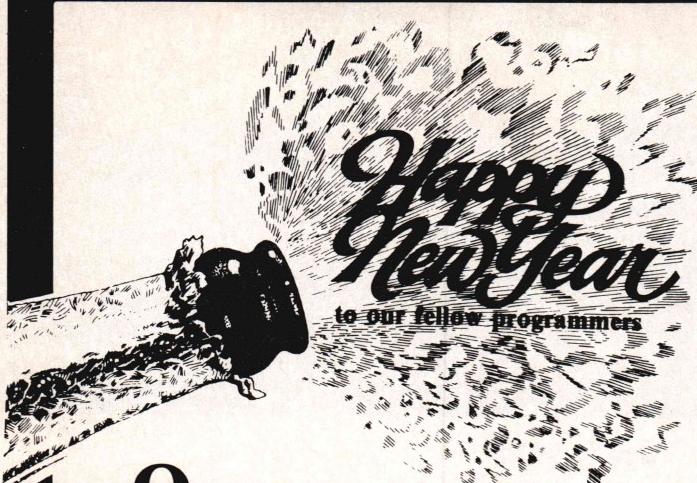
## *Listing One*

```

PGROUP GROUP PROG
PROG SEGMENT BYTE PUBLIC 'PROG'
PUBLIC ORLINE,ORPT
ASSUME CS:PGROUP
;
;-----;
; orline.asm
;
;-----;
; ROUTINE TO OR A LINE ONTO MEDIUM RESOLUTION SCREEN
; uses Bresenham's algorithm
;
orline proc near
    push bp           ; save calling bp
    push ds           ; only reg needed to save for 'C'
    push ds           ; save ds
; -----
; get x & y values
    mov si,OFFSET y2 ; addr y2
    lodsw             ; ax = y2
    xchg ax,dx        ; dx = y2
    lodsw             ; ax = x2
    xchg ax,di        ; di = x2
    lodsw             ; ax = y1
    xchg ax,cx        ; cx = y1
    lodsw             ; ax = x1
    mov bx,si          ; bx = addr color
    xchg ax,si        ; si = x1
;
    cmp si,di         ; cmp x1,x2
    jle swapxy        ; skip if (x1<=x2)
    xchg cx,dx        ; (x1>x2): swap y1,y2
    xchg si,di        ; swap x1,x2
swapxy:
;
;-----;
; | H | L |
;-----;
; ax |           |
; bx |     addr color   |
;-----;
; cx |     0 | y1 |           | if ( x1 > x2 )
;-----;
; dx |     0 | y2 |           | swap (x1,y1) with
;-----;
; si |           x1 |           | (x2,y2)
;-----;
; di |           x2 |           | i.e.
;-----;
; bp |           |
;-----;
;
; ch = deldy = (y1>y2) ? -80 : 80
; dx = |y2-y1|
    sub dx,cx          ; y2-y1
    mov al,80            ; deldy = 80
    jge ydown           ; skip if (y1<=y2)
    neg dx              ; |y2-y1|
    neg al              ; deldy = -1
ydown:
    sub di,si           ; x2-x1
; di = |x2-x1|
;
```

(Continued on page 115)

# PROGRAMMER'S DEVELOPMENT TOOLS



Over the past year we at Programmer's Connection have had the good fortune to meet and talk with hundreds of fellow programmers throughout the country. We want to extend our best wishes for a prosperous and productive New Year. And we invite you to call us for the software development tools you need in 1985.

When you call Programmer's Connection you will talk with an experienced programmer. Our people have many years of professional experience encompassing the design, programming, documentation and support of software systems ranging from large mainframes to personal computers. Hence our motto, "Programmers Serving Programmers."

Our philosophy is to offer knowledgeable and dependable service at the lowest possible price. Compare our ad with the competition . . . you'll see the difference. Call us today . . . you'll hear the difference.

Sincerely,

*Michael P. Colarik*  
Michael P. Colarik  
President,  
Programmer's Connection

**Best Wishes for 1985!**



**Programmer's Connection**

136 Sunnyside Street  
Hartville, Ohio 44632 (216) 877-3781 (In Ohio)  
"Programmers Serving Programmers"

## C LANGUAGE:

|                                       | List Ours |
|---------------------------------------|-----------|
| Computer Innovations C-86 .....       | 395 309   |
| DeSmet C Compiler with Debugger ..... | 159 145   |
| Lattice C Compiler .....              | 500 299   |
| Wizard C .....                        | 450 409   |
| Xenix Development System by SCO ..    | 1350 1099 |

### \*\*\* Mark Williams C Compiler \*\*\*

**Sale Priced at \$429**

This highly optimized compiler includes an extremely useful source-level debugger which can save you hours of programming time. A truly professional C development system with too many features to list.

## OTHER LANGUAGES:

| 8088 Assembler w/Z-80 Translator     | 100 89  |
|--------------------------------------|---------|
| BetterBASIC by Summit Software Sale! | 200 149 |
| Janus/ADA + Tools by R&R .....       | 700 499 |
| Modula-2/86 by Logitech .....        | 495 439 |
| STSC APL*Plus/PC We Can Support You! | 595 499 |

### \* Morgan Computing Professional BASIC \*

**New low price \$89**

This structured BASIC semi-compiler with debugger supports large workspaces and many extensions. 8087 support \$45.

## UTILITIES:

|   |         |
|---|---------|
| CodeSmith-86 by Visual Age .....                | 145 129 |
| Communications Library by Greenleaf .....       | 160 139 |
| C-Tree by Faircom .....                         | 395 359 |
| C To dBase by Computer Innovations .....        | 150 139 |
| C Utility Library by Essential Soft. ....       | 149 129 |
| ESP for C by Bellesoft .....                    | 349 319 |
| FirsTime for C by Spruce Tech .....             | 295 269 |
| GraphiC from Scientific Endeavors .....         | 195 169 |
| Greenleaf Functions Library .....               | 175 159 |
| Halo Color Graphics .....                       | 200 125 |
| Panel Screen Design/Editing by Roundhill .....  | 295 234 |
| Periscope Debugger by Data Base Decisions ..... | 295 269 |
| Phact by Phact Associates .....                 | 395 359 |
| Plink-86 Overlay Linkage Editor .....           | 395 310 |
| Pmate by Phoenix Software .....                 | 225 175 |
| Profiler by DWB & Associates .....              | 125 99  |
| Screen Sculptor by Software Bottling .....      | 125 109 |
| Windows For C by Creative Solutions ..          | 195 159 |

### \*\*\* Scroll & Recall \*\*\*

**\$49**

**Save \$20** on this DOS-resident Screen and Keyboard utility. S & R allows you to recall text that has scrolled off the screen. You can also recall and edit all of your previously entered DOS commands.

Prices are subject to change without notice.

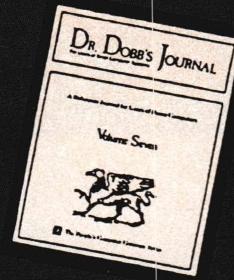
**Call for our New Catalog consisting of 200+ Programmer's Development Tools Exclusively for IBM PC's and Compatibles.**

Account is charged when order is shipped.

**1-800-336-1166**  

# Announcing BOUND VOLUME 7

Every 1982 Issue Available For Your Personal Reference.



## Vol. 7 1982

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler, and we continued to publish utility programs and useful algorithms. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi supplied a year of "Clinic" columns while delivering his famous review of JRT Pascal and writing the first serious technical comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

## Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming which was then the only way to do things. This is always pertinent for bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing.

Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI, and more.

## Vol. 2 1977

1977 found DDJ still on the forefront. These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 8080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL.

Articles are about Lawrence Livermore Lab's BASIC, Alpha-Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities, and even more.

## Vol. 3 1978

The microcomputer industry entered its adolescence in 1978. This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today.

Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors, and much, much more.

## Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages. Innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/Z80, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference.

Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe—more than ever!

## Vol. 5 1980

All the ground-breaking issues from 1980 in one volume! Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a topic of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full! Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler and, as always, even more!

## Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features.

Highlights: information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

**YES!**  Please send me the following Volumes of **Dr. Dobb's Journal**.  
 ALL 7 for ONLY \$165, a savings of over 15%!

Payment must accompany your order.

Please charge my:  Visa  MasterCard  American Express  
I enclose  Check/money order

Card # \_\_\_\_\_ Expiration Date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_ Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

**Mail to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303**  
Allow 6-9 weeks for delivery.

This offer expires February 28, 1985

Vol. 1 \_\_\_\_\_ x \$26.75 = \_\_\_\_\_  
Vol. 2 \_\_\_\_\_ x \$27.75 = \_\_\_\_\_  
Vol. 3 \_\_\_\_\_ x \$27.75 = \_\_\_\_\_  
Vol. 4 \_\_\_\_\_ x \$27.75 = \_\_\_\_\_  
Vol. 5 \_\_\_\_\_ x \$27.75 = \_\_\_\_\_  
Vol. 6 \_\_\_\_\_ x \$27.75 = \_\_\_\_\_  
Vol. 7 \_\_\_\_\_ x \$30.75 = \_\_\_\_\_  
All 7 \_\_\_\_\_ x \$165.00 = \_\_\_\_\_

Sub-total \$ \_\_\_\_\_

**Postage & Handling \_\_\_\_\_  
Must be included with order.**

Please add \$1.25 per book in U.S.  
(\$3.25 each surface mail  
outside U.S. Foreign Airmail  
rates available on request.)

**TOTAL \$ \_\_\_\_\_**

**Listing One**

```

;          H          L
;-----|-----|
; ax |           (y1>y2)? -80: 80 | al=80
;-----|-----|
; bx |           addr color | dx-cx = y2-y1
;-----|-----|
; cx |           0           y1 | if neg (y1>y2)
;-----|-----|   neg dx =|y2-y1|
; dx |           0           absdy = |y2-y1| al=-80
;-----|-----|
; si |           x1          | deldy = al
;-----|-----|
; di |           absdx = x2-x1 | di-si
;-----|-----| = |x2-x1|
; bp |
;-----|-----|
;
;         H          L
;-----|-----|
; cmp      di,dx      ; absdx,absdy
; lahf
; jnl      minmax     ; skip if (absdx>=absdy)
; xchg      di,dx
minmax:
; dx = dmin
; di = dmax
;
;          H          L
;-----|-----|
; ax | flags absdx,absdy| deldy | if (absdx < absdy)
;-----|-----| cmp di,dx lahf
; bx |           addr color | jnl -
;-----|-----| swap(absdx,absdy)
; cx |           0           y1 | xchg di,dx
;-----|-----|
; dx |           0           dmin | -:
;-----|-----|
; si |           x1          | dmax
;-----|-----|
; di |           dmax        | 
; bp |
;-----|-----|
;
;         H          L
;-----|-----|
; xchg      ax,bp      ; bp=flags(absdx,absdy) & deldy
;
; ROUTINE TO FIND INITIAL Y-ADDR, X-ADDR, AND ROTATED COLOR
;
; multiply y-coord by bytes per row and adjust for even/odd lines
    ror      cl,1      ; adjust odd/even
    mov      ax,cx      ; ax = cx = adj y-coord
    and      al,7Fh     ; page mask
    sal      cx,1      ; times 2
    sal      cx,1      ; times 4
    add      cx,ax      ; y-coord times 5
    sal      cx,1      ; times 10
    sal      cx,1      ; times 20
    sal      cx,1      ; times 40
    sal      cx,1      ; times 80
;
;          H          L
;-----|-----|
; ax |           0           | 
;-----|-----|

```

(Continued on next page)

***Listing One***

```

; bx |      addr color = addr ctableh-1      |
;-----|
; cx |          y-addr                         |
;-----|
; dx |      0      |      dmin                 |
;-----|
; si |          xl                            |
;-----|
; di |          dmax                           |
;-----|
; bp | flags absdx,absdy|      deldy           |
;-----|
;

; compute the rotated mask and color
; bx = ctableh-1 = addr color
    mov     al,3          ; pixel position mask
    and     ax,si         ; just the bit count into the index
    add     al,[bx]        ; pixel position + color (* 4 + 1)
    xlat              ; look up the masks al=[al+bx]
;

    mov     bx,0B800H      ; disp seg base addr
    mov     ds,bx          ; ds = display base addr
;

; al = rotated color
; cx = y-addr offset
; ds = display addr
;

; .. H .. L ..
; ax |      0      |      rotated color       |
;-----|
; bx |          |                                |
;-----|
; cx |          y-addr                         |
;-----|
; dx |      0      |      dmin                 |
;-----|
; si |          xl                            |
;-----|
; di |          dmax                           |
;-----|
; bp | flags absdx,absdy|      deldy           |
;-----|
;

    sal     dx,1          ; dx = delse = dmin * 2
    xchg   cx,di          ; cx = dmax, di = y-addr
    xchg   ax,dx          ; ax=delse, dx=rotated color
    xchg   ax,bp          ; ax=flags(absdx,absdy), bp=delse
;

; .. H .. L ..
; ax | flags absdx,absdy|      deldy           |
;-----|
; bx |          |                                |
;-----|
; cx |          dmax                           |
;-----|
; dx |      0      |      rotated color       |
;-----|
; si |          xl                            |
;-----|
; di |          y-addr                         |
;-----|
; bp |      dmin * 2 = delse                  |
;-----|
;
```

```

;
; sahf          ; cmp absdx,absdy
pushf

;
cbw           ; ax = deldy
mov  cs:delsy,ax ; save deldy
mov  cs:deldy,ax
mov  cs:deldy2,ax
mov  bx,di      ; bx = y-addr
or   ax,ax
js   negdeldy   ; if deldy<0 jmp

;
test bh,20H    ; is page bit set?
jz   toggle     ; skip to toggle page

;
add  bx,ax      ; add deldy to y-addr
jmp  toggle     ; skip to toggle page

negdeldy:
test bh,20H    ; is page bit set?
jnz  toggle     ; skip to toggle page

;
add  bx,ax      ; add deldy to y-addr
toggle:
xor  bh,20H    ; flip page bit
bx = next y-addr
;

.....| H | L |
-----+---+---+
; ax |
; bx | y-addr (next) |
; cx | dmax |
; dx | 0 | rotated color |
; si | xl |
; di | y-addr |
; bp | dmin * 2 = delse |
;

xchg ax,bp      ; ax = dmin * 2 = delse
mov  cs:delse,ax ; save delse
mov  cs:delse2,ax
sub  ax,cx      ; ax = dmin * 2 - dmax = d
mov  bp,ax      ; bp = d = error term
sub  ax,cx      ; ax = dmin * 2 - dmax * 2
mov  cs:delde,ax ; save delde
mov  cs:delde2,ax ; save delde

;
xchg ax,bx      ; ax = next y-addr

;
figure x-coord address
mov  bx,si      ; get x-coordinate
sar  bx,1       ; divide
sar  bx,1       ; by 4
bx = x-addr offset
;

.....| H | L |
-----+---+---+
; ax | y-addr (next) |
; bx | x-addr |
; cx | loop count |
;

```

(Continued on next page)

# PROFESSIONAL BASIC™

JUST BECAME AFFORDABLE

**\$99**

"outstanding" Ray Duncan - Dr. Dobbs' Journal

Use 640k (e.g. 250 x 250 array)

Dynamic Syntax Checking

19 Debugging Windows

Run PC BASICA Programs

OPTIONAL: 8087/80287 Support — \$50



Morgan Computing Co., Inc.  
(214) 739-5895

10400 N. Central Expwy., Suite 210  
Dallas, TX 75231

Circle no. 51 on reader service card.

## NOW C HERE! CROSS SOFTWARE for the NS32000

Also Available for IBM PC  
INCLUDES:

- \* Cross Assembler \*
- \* Cross Linker \*
- \* Debugger \*
- \* N.S. ISE Support \*
- \* Librarian \*
- \* Pascal Cross Compiler \*
- \* C Cross Compiler \*

U.S. prices start at \$500

**SOLUTIONWARE**  
1283 Mt. View-Alviso Rd.  
Suite B  
Sunnyvale, Calif. 94089  
408/745-7818 • TLX 4994264

Circle no. 96 on reader service card.

## ICs PROMPT DELIVERY!!! SAME DAY SHIPPING (USUALLY)

8087-3 Co-Processors \$131.97

### DYNAMIC RAM

|      |        |        |         |
|------|--------|--------|---------|
| 256K | 256Kx1 | 150 ns | \$17.47 |
| 64K  | 64Kx1  | 120 ns | 3.77    |
| 64K  | 64Kx1  | 150 ns | 2.97    |
| 64K  | 64Kx1  | 200 ns | 3.07    |

### EPROM

|       |       |        |         |
|-------|-------|--------|---------|
| 27256 | 32Kx8 | 300 ns | \$37.50 |
| 27128 | 16Kx8 | 250 ns | 13.57   |
| 27C64 | 8Kx8  | 200 ns | 12.47   |

### STATIC RAM

|       |      |        |      |
|-------|------|--------|------|
| 2764  | 8Kx8 | 250 ns | 6.50 |
| 2732A | 4Kx8 | 250 ns | 6.37 |
| 2716  | 2Kx8 | 450 ns | 3.31 |

### QUANTITY ONE PRICES SHOWN

|          |      |        |         |
|----------|------|--------|---------|
| 6264P-15 | 8Kx8 | 150 ns | \$19.57 |
| 6116P-3  | 2Kx8 | 150 ns | 4.37    |

Open 6½ days: We can ship via Fed-Ex on Sat.

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts

MICROPROCESSORS UNLIMITED  
24,000 South Peoria Ave. (918) 267-4961  
BEGGS, OK. 74421

Prices shown above are for November 26, 1984

Please call for current prices & volume discount. Prices subject to change. Please expect higher or lower prices due to market conditions. Prices do not include shipping, handling, taxes, and insurance extra. Cash discount prices shown. Small orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ \$6.75!

Circle no. 64 on reader service card.

***Listing One***

```

; dx |           | rotated color |
;-----|           |-----|
; si |           x value |
;-----|           |-----|
; di |           y-addr |
;-----|           |-----|
; bp |           error term |
;-----|           |-----|
;
;
;          popf      ; cmp absdx,absdy
;          jns       delsx2    ; if (absdx>=absdy) goto delsx2
;
;-----|
; delsx = 0  (absdx < absdy)
;
        or      [bx][di],dl   ; or disp with color (plot point)
        jcxz   lineexit     ; quit when cx=0
        or      bp,bp        ; set bp flags
        jge    diagonal      ; if bp>=0 jmp
;
; case for straight move
straight:
        xchg   ax,di        ; every other for page adj
delsy  = $+1
        add    ax,fakedw    ; ++y
;
        or      [bx][di],dl   ; or disp with color (plot point)
        dec    cx            ; --loop counter
        jz     lineexit     ; quit when cx=0
;
delse  = $+2
        add    bp,fakedw    ; update error term
        js    straight      ; if bp<0 goto straight
;
; case for diagonal move
diagonal:
        inc    si            ; ++x value
        mov    bx,si        ; bx = x value
        sar    bx,1          ; bx = x addr offset
;
        ror    dl,1          ; adjust color position
        ror    dl,1
;
        xchg   ax,di        ; every other for page adj
deldy  = $+1
        add    ax,fakedw    ; ++y
;
        or      [bx][di],dl   ; or disp with color (plot point)
        dec    cx            ; --loop counter
        jz     lineexit     ; quit when cx=0
;
delde  = $+2
        add    bp,fakedw    ; update error term
        js    straight      ; if bp<0 goto straight
        jmp    diagonal      ; if bp>=0 goto diagonal
;
;-----|
delsx2:
; delsx = 1  (absdx >= absdy)
;
        or      [bx][di],dl   ; or disp with color (plot point)
        jcxz   lineexit     ; quit when cx=0
        or      bp,bp        ; set bp flags
        jge    diagonal2     ; if bp>=0 jmp
;
```

(Continued on page 120)



***Listing One***

```

; case for straight move
straight2:
    inc    si           ; ++x value
    mov    bx,si         ; bx = x value
    sar    bx,1
    sar    bx,1         ; bx = x addr offset
;
    ror    dl,1          ; adjust color position
    ror    dl,1
;
    or     [bx][di],dl   ; or disp with color (plot point)
    dec    cx           ; --loop counter
    jz     lineexit      ; quit when cx=0
;
delse2 = $+2
    add    bp,fakedw    ; update error term
    js    straight2      ; if bp<0 goto straight
;
; case for diagonal move
diagonal2:
    inc    si           ; ++x value
    mov    bx,si         ; bx = x value
    sar    bx,1
    sar    bx,1         ; bx = x addr offset
;
    ror    dl,1          ; adjust color position
    ror    dl,1
;
    xchg  ax,di         ; every other for page adj
deldy2 = $+1
    add    ax,fakedw    ; ++y
;
    or     [bx][di],dl   ; or disp with color (plot point)
    dec    cx           ; --loop counter
    jz     lineexit      ; quit when cx=0
;
delde2 = $+2
    add    bp,fakedw    ; update error term
    js    straight2      ; if bp<0 goto straight
    jmp    diagonal2      ; if bp>=0 goto diagonal
;
; -----
lineexit:
    pop    ds           ; restore ds
    pop    bp           ; restore calling bp
    ret
;
orline endp
;
; -----
; orpt.asm
;
; -----
; ROUTINE TO OR A POINT ONTO MEDIUM RES COLOR SCREEN
;
orpt  proc  near
;
; get initial values for x and y
    mov    si,OFFSET yl   ; addr yl
    lodsw             ; ax = yl
;
; multiply y-coord by bytes per row and adjust for even/odd lines
    ror    al,1          ; adjust odd/even
;
    mov    dx,0B87FH      ; disp addr and page mask

```

(Continued on page 122)

# \$3.00 C Compiler

Due to popular demand, **Dr. Dobb's Journal** has reprinted its most-asked-for C compiler articles by Ron Cain and J. E. Hendrix, each for only \$3.00.

Ron Cain's C compiler from sold-out 1980 issues #45 and #48 includes "A Small C Compiler for the 8080s" and "Runtime Library for the Small C Compiler."

The J. E. Hendrix reprint includes part two of "Small-C Compiler v.2" from sold out issue #75 and completes the first part of the compiler article from issue #74 which is included in Dr. Dobb's Bound Volume 7.

To Order: Enclose \$3.00 for each copy with this coupon and send to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Please send \_\_\_\_\_ copy(ies) of the Ron Cain Reprint, and  
\_\_\_\_\_ copy(ies) of the J. E. Hendrix reprint to:

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

**ALL REPRINT ORDERS MUST BE PREPAID.**

99

Circle no. 83 on reader service card.



## TOTAL CONTROL WITH WL FORTH

**PC FORTH™**  
**IBM PC & XT,**  
**HP-150,**  
**Macintosh,**  
**Apple II,**  
**CompuPro,**  
**Sage & CP/M-68K,**  
**Wang PC,**  
All CP/M and  
MSDOS computers.

Try the professional language offering the utmost performance in the shortest development time. Transport your applications between any of our enhanced 83-Standard compilers or expanded 32-bit versions. Choose from our wide selection of programming tools including native code compilers, cross-compilers, math coprocessor support, and B-Tree file managers. All fully supported with hotline, updates, and newsletters.

**Laboratory Microsystems Incorporated**

Post Office Box 10430, Marina del Rey, CA 90295  
Phone credit card orders to (213) 306-7412



Circle no. 55 on reader service card.

## 33 KFLOPS

Use your IBM PC, XT or AT to multiply two 128 by 128 matrices at the rate of 33 thousand floating-point operations per second (kflops)! Calculate the mean and standard deviation of 16,384 points of single precision (4 byte) floating-point data in 1.4 seconds (35 kflops). Perform the fast Fourier transform on 1024 points of real data in 6.5 seconds. Near PDP-11/70 performance when running the compute intensive Owen benchmark.

### WL FORTH-79

FORTH-79 by WL Computer Systems is a powerful and comprehensive programming system which runs on the IBM PC (and some compatibles) under PC DOS 1.1 or later. If your computer has the 8087 numeric data processing chip (NDP) installed, then this version of FORTH-79 will unleash the awesome floating-point processing power which is present in your system. If you haven't gotten around to installing the 8087 NDP coprocessor in your computer, you can still use WL FORTH to write applications using standard FORTH-79.

System includes editor, memory dump, decompiler, nondestructive stack printout, screen printer and screen copy utilities. FORTH sources for these utilities are included.

Unlike most other products, the **complete** source is available at a very affordable price.

Package 1 includes FORTH-79 versions with and without 8087 support. Included are screen utilities, 8087 and 8088 FORTH assemblers. \$100

Package 2 includes package 1 plus the assembly language source for the WL FORTH-79 nucleus. \$150

Package 3 includes package 2 plus the WL FORTH-79 source screens used to add the 8087 features to the vocabulary. \$200

Starting FORTH book. \$22

**WL Computer Systems**  
1910 Newman Road  
W. Lafayette, IN 47906  
(317) 743-8484

Visa and Master Card accepted.

IBM is a trademark of International Business Machines

Circle no. 110 on reader service card.

***Listing One***

```

and    dl,al          ; mask page bit, disp + y.coord
sal    ax,1           ; times 32
sal    ax,1           ; times 64
add    dx,ax          ; addr disp seg + y-coord times 5 (80)

; compute x-coord address offset
lodsw              ; ax = xl
mov    di,ax          ; get x-coordinate
sar    di,1           ; divide
sar    di,1           ; by 4

; compute the rotated mask and color
and    al,3           ; just the bit count into the index
add    al,[si]         ; pixel position + color (* 4 + 1)
mov    bx,si          ; bx = ctableh - 1
mov    si,ds          ; save ds
xlat              ; look up the masks al=[al+bx]

; mov    ds,dx          ; set seg to disp + y-addr
; or     [di],al         ; or the byte with the color
; mov    ds,si          ; restore ds
; ret

;orpt   endp
;

;-----
```

PROG ENDS  
END

End Listing One

***Listing Two***

```

/*
This is an example written in 'C' using the line drawing routine.
-- Bruce Smith.
*/

extern int xl,y1,x2,y2;
extern char color;

main()
{
int incr=3;
pcvsym(4); /* color med res graphics */

color=(2<<2)+1;

xl=160; y1=100; x2=0; y2=0;

for (; x2<319; x2+=incr) orline(); /* draw line */
x2=319;

for (; y2<199; y2+=incr) orline(); /* draw line */
y2=199;

for (; x2>0; x2-=incr) orline(); /* draw line */
x2=0;

for (; y2>0; y2-=incr) orline(); /* draw line */
cget(); /* wait for keypress */
pcvsym(2); /* 80 col b & w */
}
```

End Listing Two

## Now With Windowing! \$49.95 Basic Compiler

**MTBASIC**

### Features:

|                    |                  |
|--------------------|------------------|
| Multitasking       | Windowing        |
| Handles interrupts | Interactive      |
| Fast native code   | Compiles quickly |
| Floating point     | No runtime fee   |

MTBASIC is a true native code compiler. It runs Bytes's Sept. '81 sieve in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

MTBASIC combines the best of interpreters and compilers. To the programmer, MTBASIC appears to be an extremely fast interpreter. MTBASIC compiles a typical 100 line Basic program in 1 second, yet it generates blindingly fast code. No more waiting for long compiles.

AVAILABLE FOR CP/M (Z-80) and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send \$49.95 plus \$3.50 shipping and handling (\$10 overseas) to:

**SOFIAID, Inc.**

P.O. Box 2412 Columbia, MD 21045-1412  
301/792-8096

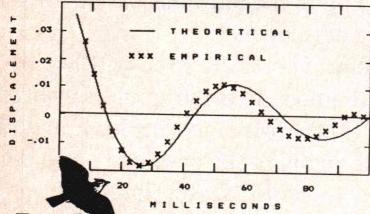
Circle no. 88 on reader service card.

## Graphs without Graphics?

No need for screen graphics. Publishable graphs on your dot matrix printer.

Easy to Use. No programming.  
CP/M 80 or 86, MS-DOS, or PC-DOS.  
Excellent Manual. Most disk formats.

## DataPlotter™



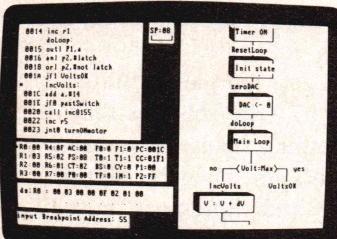
**Lark Software™**  
7 Cedars Road  
Caldwell, NJ 07006

Line Graphs &  
Scatterplots . . . . \$69  
Bar Graphs &  
Pie Charts . . . . \$69  
Both for . . . . \$99  
(Prices include manual)

(201) 226-7552

Circle no. 57 on reader service card.

## Simulator/Debugger for IBM-PC for 8048 - 8051 - 7000 µCs



Execute and debug code for popular single chip microcomputers on your IBM PC. Dynamic display with windows for source code, control flow, registers, flags, memory, commands, and more! Set breakpoints, traps, etc. Cross-assembler & EPROM programmers, too.

| X-Assembler | ✓ | ✓ | ✓ | ✓ | \$195 |
|-------------|---|---|---|---|-------|
| EPROM Prog  | ✓ | ✓ | ✓ | ✓ | \$245 |
| Simulator   | ✓ | ✓ | ✓ |   | \$395 |

**Cybernetic Micro Systems**  
Box 3000 • San Gregorio, California 94074 USA  
(415) 726-3000 • Telex 171-135 Attn: Cybernetics

Circle no. 23 on reader service card.

A general purpose programming language for string and list processing and all forms of non-numerical computation.

## SNOBOL4+

— the entire SNOBOL4 language with its superb pattern-matching facilities • Strings over 32,000 bytes in length • Integer and floating point using 8087 or supplied emulator

• ASCII, binary, sequential, and random-access I/O • Assembly Language interface • Compile new code during program execution • Create

SAVE files • Program and data space up to 300K bytes

RAM

With ELIZA & over 100 sample programs and functions

For all 8086/88 PC/MS-DOS or CP/M-86 systems, 128K minimum 5 1/4" DSDD, specify DOS/CPM format

Send check, VISA, M/C to: \$95

Catspaw, Inc. plus \$3 s/h

P.O. Box 1123 • Salida, CO 81201 • 303/539-3844

Circle no. 20 on reader service card.

## cVIEW™ SCREEN MANAGER

cVIEW helps to create and communicate with forms that will enable you to provide a sophisticated user interface for your C applications programs.



**CI86, Lattice, Mark Williams Dynamic Forms Position Special Key Definitions Input Range Validation Color or Monochrome Block Cursor Option Selection Sets**

cVIEW Screen Manager ..... \$245.00  
cVIEW with Source Code ..... \$545.00  
cVIEW Demo ..... \$25.00

**CompuCraft (313)**  
42101 Mound Road 731-2780  
Sterling Heights, Michigan 48078

Circle no. 9 on reader service card.

## FORTH

For 65SC802  
The 16 bit 6502

Apple, Atari, AIM-65 Commodore, OSI

Faster 16 bit code

Starlight FORTH Systems  
15247 N. 35th St.  
Phoenix, 85032

Also  
FORTH tools  
For all FORTH systems

Circle no. 67 on reader service card.

## BIG DISCOUNTS ON LITTLE BOARDS™ & ACCESSORIES



- **AMPRO LITTLE BOARD™** — 64K, 280a CPU, CTC, DART, 1 parallel port, 5 1/4 controller supports four 48pipt and/or 96pipt drives w/ CP/M 2.2 and ZCPR3 (A & T) . . . . . from \$329
- **SYSTEM SUPPORT PKG** — Manuals, source code, schematics, connectors & cables . . . . . \$99
- **SCSI PLUS** — DMA Hard disk interface . . . . . \$99
- **TEAC 55B DSDD** 48pipt 1/2 ht drive . . . . . \$195
- **TEAC 55D DSDD** 96pipt 1/2 ht drive . . . . . \$239
- **INTEGRAND** Custom two drive cabinet with 5 amp power supply & power cables . . . . . \$199
- **TERM-MATE** — Cabinet for 2 1/2 ht + LITTLE BOARD w/ all cables & supply . . . . . \$229
- **AMPRO SERIES 100** complete systems . . . . . SCALL

VISA & MASTER CHARGE. Personal Checks.

Please allow 2 weeks. Shipped via UPS.

Prices F.O.B. Prairie View, IL.

For additional information write or call:

DISKS PLUS • 15945 West Pope Blvd. • Prairie View, IL 60069

(312) 537-7888

**DISKS PLUS**  
DIVISION OF SOLARONICS INC

Circle no. 34 on reader service card.

## No source code for your REL files?

## REL/MAC

converts a REL file in the Microsoft™ M80 format to a ZILOG™ or 8080 source code MAC file with insertion of all public and external symbols.

- REL/MOD lists library modules
- REL/VUE displays the bit stream
- 50 page manual with examples
- free brochure available
- REL/PAK includes all of the above

REL/PAK for 8080 only ..... \$99.95  
REL/PAK for Z80 & 8080 ..... \$134.95  
on 8'SSSD disk for CP/M™ 2.2

Send check, VISA, MC or C.O.D. to



**MICROSMITH**  
COMPUTER TECHNOLOGY

P.O. BOX 1473 ELKHART, IN 46515

1-800-622-4070

(Illinois only 1-800-942-7317)

## C Programmers B-Trees

**\$75.00** + 2.00 Postage

### Source Code Included

The Softfocus B-Trees record indexing library will help you develop sophisticated application programs. With Softfocus B-Trees you get:

- high speed file handling for up to 16.7 million records per file
- customizable BDS or K & R standard C source code
- no royalties on applications programs
- support random and sequential file access
- includes example programs

**Softfocus**

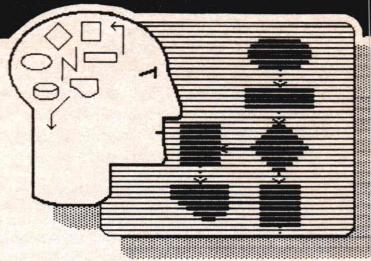
1277 Pallantine Dr., Oakville, Ont.  
Canada L6H 1Z1 (416) 844-2610



Circle no. 23 on reader service card.

Circle no. 41 on reader service card.

Circle no. 89 on reader service card.



by Michael Swaine

Tom Evaslin and his company, Solutions Inc., have written seven programs for the Macintosh so far, including Dow Jones Straight Talk and Spreadsheet Link, Rags to Riches Payable and Receivables, and learning games for Que. Eighty percent of the company's development work is software for the Mac. As the first person in his company to do Mac development, Evaslin has some opinions about what it's like to develop software for the machine named after a raincoat.

**Evaslin:** As everyone has been saying, the Mac presents a bigger hurdle to software development than any past microcomputer. The vocabulary of capabilities represented by those utilities in ROM is powerful, but the Mac gives you no easy way to start using them. Traditionally, when you were going to write a program for an Apple computer, you started by getting into BASIC and writing a Hello program. Well, there's no way to do that on the Mac. There is no immediately obvious way to write to the screen, no obvious way to grab keyboard input. This was intentional; it forces you to get into and use that excellent interface. But at first, it's discouraging. But then there follows a period of euphoria. Each programmer on our team has gone through this depression, when there appear to be random events going on. Then the euphoria comes as he begins to learn how to make pulldown menus and scrollbars work.

**DDJ:** I'm sure that's a nice feeling, but what's the practical implication?

**Evaslin:** Overall, you become a more productive software developer. You just wouldn't build that kind of user interface into every product if you had to do it from scratch every time. The Mac software environment puts that interface into every product for you.

**DDJ:** What problems do you run into in porting software into the Mac

environment?

**Evaslin:** You don't port software to the Mac. Or we don't. Our products are all new code. We look at the old code; we have it to look at as we work; but what we write is all new code. You can't just translate software from another environment to the Mac; when you do, the result is a piece of junk.

**DDJ:** What kind of support do developers get from Apple in learning to work in the environment, learning to develop software for the machine? Is Mac college helpful?

**Evaslin:** Apple runs a Mac college but until recently there was no Mac high school. There was no way to get there from here. Now there are a few people providing that level of training, but most people have had to get there by the school of hard knocks. Mark Ursino (formerly of Microsoft) is one of these teaching at the lower level; Mark has taught a Mac high school at Apple. That was helpful.

**DDJ:** The user documentation for the Mac is attractive and well organized, but I was a bit surprised when I first saw the pile of photocopied pages that make up the documentation a developer gets. Isn't that documentation a handicap?

**Evaslin:** The lack of high-level, structured documentation has been a problem. Apple has given programmers one level of documentation, with no way to get an overview. You read the window management section and you will probably learn more than you would ever want to know about window management in the Mac environment, but there's no way to know, short of exhaustive search, if what you want to know about a particular point in window management is in the documentation.

**DDJ:** What about support?

**Evaslin:** There are bugs, but few of them, considering all that is in the ROM. And there is developer support.

Not every programmer gets all the access to the developers that he wants when he wants it, but contrast this with IBM, where developers are just inaccessible. Right now, you can actually talk to the people who were involved in the development of the Mac software. You can get at the actual developers.

**DDJ:** Has it been a hindrance that you have to work on the Lisa?

**Evaslin:** Having to do development on another machine is a barrier. It's not so bad for a company like ours, but it is difficult for somebody working in his basement to have to go out and buy another machine to develop software for the Mac.

**DDJ:** Do you think that the 512K Mac is going to make things easier for software developers?

**Evaslin:** The 512K Mac will not be as great a boon to programmers as everyone thinks. Memory has been overrated as an aid to the developer. You don't have to build a user interface or do those other things the ROM routines do for you, so you have more of the RAM to use anyway. 128K is not terribly limiting. What's confusing is that the Mac memory management is hard to get used to. That's really where the problems lie, not in the amount of memory. The larger machine may in fact mask bugs more easily than the 128K machine. With the 128K, the heap gets reorganized fairly often. It's easy to write routines that depend on something being relocatable when in fact it's not. Then the heap gets reorganized and things get moved and you see an erratic, nonrepeatable bug. With 512K you may never see the bug crop up in beta test.

I was the first in our company to do development on the Mac and I saw all the others going through the learning process. Nowhere did anybody have as much trouble as in memory management. Where handles get allocated,

heap management. Hardest-to-track bugs.

**DDJ:** The predictions are that Jack Tramiel at Atari and everyone else will soon be bringing out Maclike machines, and the predictions seem likely to come true, more or less. What do you think we can expect from such machines?

**Evsin:** Those who are going for the low-cost market will opt for maximum compatibility with the Mac; those selling functionality will go for merely using the conceptual interface and will try to differentiate themselves in terms of functions. Both high and low levels of compatibility will emerge.

The Mac itself is poised to penetrate two new markets: people who would never have bought a computer before and the professional market. The former, for sure, as more software is developed that is fun. We will see not so much a fight with IBM for market share as an expansion of the market. Professionals, I'm not so convinced about, but as the so-called power software comes out, the opportunity will exist.

[After we spoke, Apple leaked its current marketing plans: the Christmas "test drive a Mac" deal and the plan to go after corporate middle managers, leaning on the new Lotus product for the Mac, the planned Mac local-area network, a laser printer, and a telephone integrated into the machine. CEO John Sculley claimed that Apple has a two-year window for getting Macs on corporate desktops and said that the plan was not one that puts Apple on a collision course with IBM.]

DDJ

**Reader Ballot**

Vote for your favorite feature/article.  
Circle Reader Service No. 197.

# 3,500 Programmers depend on us to find, compare, evaluate products and for solid value.

THE PROGRAMMER'S SHOP serves serious microcomputer programmers . . . from giant institutions to small independents. Specializing helps us provide 100s of programming products . . . technical literature . . . specialized evaluations and more to help you find and evaluate. Other services like . . . special formats . . . rush delivery . . . payment options (POs, COD, credit cards, etc.) . . . newsletters . . . and reports help you save time, money, and frustration and get solid value.

## ARTIFICIAL INTELLIGENCE

EXSYS - Expert System building tool. Full RAM, Probability, Why, Intriguing, serious. PCDOS \$200

GC LISP - "COMMON LISP", Help, tutorial, co-routines, compiled functions, thorough. PCDOS \$475

TLC LISP - "LISP-machine"-like, all RAM, classes, turtle graphics 8087 for CP/M-86, PCDOS or MSDOS \$235

TLC LOGO - fast, classes. CPM \$139

PROLOG-86 - Learn fast, Standard, tutorials, samples of Natural Language, Exp. Sys. MSDOS \$125

Expert System front-ends for PROLOG: APES (\$275), ES/P (\$1895)

Other solid alternatives include:  
IQ LISP (\$155), MuLISP-86  
\$250, WALTZ LISP for CPM  
(\$159), MicroPROLOG (\$275).

## DEBUGGERS

CODESMITH-86 - Symbolic, multi-window, very visual. PCDOS \$139

PERISCOPE DEBUGGER - load after "bombs", symbolic, "Reset box", 2 Screen, own 16K. PCDOS \$295

Consider others like SYMD (\$119), Mylstar (\$119), Pfix (\$175), TRACE (\$115), ATRON (\$269).

ACTIVE TRACE for BASICA, MSBASIC, CMP or MSDOS. \$72

SOURCE PROBE by Atron for Lattice, MS C, Pascal. Windows single step, 2 screen, log file. PCDOS \$395

## C PROGRAMMING

INSTANT C - Interactive development - Edit, Source Debug, run. Edit to Run - 3 Secs. MSDOS \$500

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial, examples, graphics. PCDOS \$95

MEGAMAX C - native Macintosh has fast compile, tight code, K&R, toolkit, .OBJ, DisASM MAC \$295

CROSS COMPILERS by Lattice, CI. VAX to 8086. VMS \$3000

## C LIBRARIES

COMMUNICATIONS by Greenleaf (\$149) or Software Horizons (\$139) includes Modem7, interrupts, etc. Source.

C SHARP Realtime Toolkit - well supported, thorough, portable, objects, state sys. Source \$600

GRAPHIC - scientific graphics, 4096 by 3200 pixel - map to device, Zoom. 8087, Source. MSDOS \$195

ROMPack - special \$Main .EXE editor, source, tech support, 8086. \$195

## SUPPORT PRODUCTS

BRIEF Programmer's Editor - undo, windows, reconfigurable, macro programs, powerful. PCDOS \$195

PLINK-86 for Overlays, most lang., segment control. MSDOS \$325

PROFILER-86 - faster programs with less work. Learn quick, symbolic, All Lang. MSDOS \$125

SCIL - Source Librarian to manage Versions, Doc, Minimize disk space, confusion. MSDOS \$335

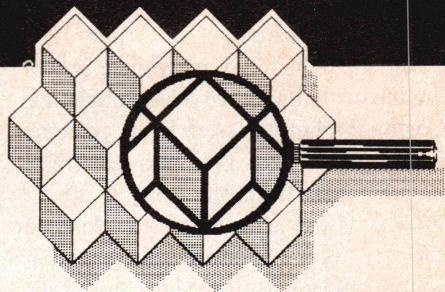
Call for a Catalog, literature on any product or a free literature "Packet" on: "AI", BASIC, C, COBOL, Debuggers, Editors, FORTH, FORTRAN, Libraries, PASCAL

**CALL TOLL FREE 800-421-8006**

# THE PROGRAMMER'S SHOP™

*The programmer's complete source for software, services and answers*

128-D Rockland Street, Hanover, MA 02339 In Mass.: 800-442-8070 or 617-826-7531



by R. P. Sutherland

## Subliminal Software

New Life Institute has released software for the IBM PC and compatibles designed to reprogram the user's habits, behaviors, and personal beliefs. The program entitled **Subliminal Suggestions and Self Hypnosis Programs for Your Computer!** flashes user-chosen messages as a super-high-speed subliminal background on the screen. The subliminal messages can be used to reprogram the user's subconscious. The package includes separate programs for self-hypnosis and deep relaxation. The accompanying documentation contains a warning about possible misuses of subliminal technology. The "Subliminal Suggestions and Self Hypnosis" package retails for \$75.00. The programs are written in C with several assembly language routines. Contact New Life Institute, P.O. Box 2390, Santa Cruz, CA 95063 (408) 429-1122. **Reader Service No. 109.**

## Babel

The Babel that towers above Silicon Valley casts a shadow around the globe. A high-tech lingua franca could speed aspects of the microcomputer revolution, but diversity promotes creativity. If, by joining forces, the United States and Japan could produce the man-like machine that Turing predicted by the 21st century, one can't help but wonder if the present confusion is God's attempt to frustrate our progress.

I attended Gary Kildall's PC Faire presentation in San Francisco last year, in which he was supposed to prophesy the direction of the microcomputer industry over the next three years. Instead, he traced the development of microprocessors, memory, and input/output devices during the last

decade and limited any predictions to one year. Within a year, he said, programmers may assume that the standard business computer will have 3 megabytes of RAM, a CPU in the 80286 category or a 32-bit processor, and, thanks to laser disk technology, enough storage to contain the *Encyclopaedia Britannica*. All of this speed and power means that we are going to be able to get a lot more work out of the microcomputer. How? Kildall had the solution: a copy of Concurrent DOS, still in the shrink-wrap, which he offered to sell because, he said, he needed plane fare home. Concurrent DOS, like TopView, is a multitasking, windowed program to enhance MSDOS. Microsoft Windows, a similar product, has been delayed until June.

Kildall seemed most enthusiastic about **Virtual Device Interface (VDI)** technology. By raising the functionality of all devices to a common level, VDI allows one to create programs that are not at the mercy of technological leapfrogs. The VDI presents a general interface through which high-level programs can communicate with a wide variety of hardware. IBM, Digital Research Institute, and now Ashton-Tate are licensing GSS-Drivers from Graphic Software Systems. GSS-Drivers is an implementation of the American National Standards Institute's proposed VDI programming standard, joined with a library of device drivers, which permits input and output device independence. Graphic Software Systems is located at 25117 South West Parkway, Wilsonville, OR 97070. Contact: William Merchant (503) 682-1606.

Vertex Systems has announced **Apple-Turnover**, a board that gives IBM PCs the ability to read and write Apple-Dos 3.3 and Apple CP/M disks. Vertex Systems also has XenoCopy, a system

of programs to copy files between disks of different computers. I wanted to compare it to microVersal, a similar product that we have been using with mixed results, but Vertex was unable to part with a review copy. Apple-Turnover is priced at \$279.50 and XenoCopy by Fred Cisin is \$99.50 or \$149.50 for XenoCopy Plus, which permits writing to and formatting other disks. Vertex Systems, Inc., 6022 W. Pico Blvd., Los Angeles, CA 90035 (213) 938-0857. **Reader Service No. 105.**

CompuPro users may now run PCDOS at CompuPro speed. Computer House announces **PCPRO** version 2.4. PCPRO is IBM's PCDOS version 2.1 modified to operate on CompuPro letter series systems. The package includes INTERDOS, a CP/M to MSDOS transfer utility, and COFIGIO, an interactive I/O configurative utility. PCPRO: PCDOS for CompuPro costs \$395.00 from Computer House, 722 B Street, San Raphael, CA 94901 (415) 453-0865. **Reader Service No. 107.**

## Hardware

Logitech has applied infrared technology to the mouse and produced **the first cordless mouse** as a result of custom development work for Metaphor Computer Systems. Logitech Inc. is located at 805 Veterans Blvd., Redwood City, CA 94063 (415) 365-9852. **Reader Service No. 111.**

CompuPro has introduced a high-speed, low-power static RAM board compatible with both 8- and 16-bit processors. The **RAM 23** provides up to 128K of static RAM and operates at up to 12 MHz with 8086/8088/80286 CPUs. The RAM 23 has a suggested retail price of \$400.00 for the 64K version and \$775.00 for the 128K version. Contact Jeff Swartz, CompuPro, 3506 Breakwater Court, Hayward, CA

94545 (415) 786-0909. **Reader Service No. 115.**

Morrow has begun shipping a 10-pound battery-operated portable computer that operates under MSDOS and uses standard 5½-inch floppy-disk drives. A 640K dual drive version is available for \$3,695.00. Standard features include: clock, built-in 300 baud modem, calculator, and word processor. The CPU is a CMOS 80C86. The **Pivot** is available from Morrow, 600 McCormick, San Leandro, CA 94577. **Reader Service No. 117.**

## Miscellany

### Musical Software

**Tune Smith/PC** is the first software for musical composition for the IBM PC. Price: \$49.95 from Blackhawk Data Corporation, 307 N. Michigan

Avenue, Chicago, IL 60601 (312) 236-8473. **Reader Service No. 119.**

### Disk Preventive Maintenance

**Disk P.M.** from Digital Pathways, Inc., performs preventive maintenance on hard and floppy disks for IBM PCs and compatibles. Disk P.M. diagnoses problems, automatically condenses hard or floppy disks, restores order, rebuilds damaged directories, recovers damaged files, locks out faulty areas, and copies system information to disks that refuse to boot. Retail price is \$49.95 from Digital Pathways, Inc., 1060 East Meadow Circle, Palo Alto, CA 94303. **Reader Service No. 121.**

### Public Domain Help Function

This software package provides full **on-screen help** for all PCDOS version 2.0 and 2.1 commands. The actual help text may be altered by the user. Copies are available for \$10.00 from Chris

Bailey, P.O. Box 332, Peterborough, Ontario, Canada, K9J 6Z3. **Reader Service No. 125.**

### TRS-80 Screen Dump

A high-resolution **screen dump utility for TRS-80 Models I, II, III, and 4** allows users to dump the contents of a video screen, text and graphics, to an Epson or Gemini printer. The price is \$19.95, available from Softbyte Computing, Box 217, Wallingford, CT 06492 (203) 239-6923. **Reader Service No. 123.**

### Tools for Turbo Pascal

**Programming Tool Kit** by Paragon Courseware is a set of utilities for Turbo Pascal programmers. The kit includes a Window Package, a Function Evaluating Package, and a System Information Package. The Graphics Package has several procedures that draw various shapes such as triangles, parallelograms, other polygons, circles, and ellipses. The Tool Kit retails for \$49.95 from Paragon Courseware, 4954 Sun Valley Road, Del Mar, CA 92014 (619) 481-1477. **Reader Service No. 127.**

### Forth Authors

The Forth Interest Group announces the formation of its **Author Recognition Program**, which offers free FIG membership to the author of any Forth-related article published in a periodical. FIG hot line: (415) 962-8653.

The **1985 Rochester Forth Conference** will be held June 12–15 at the University of Rochester, New York. The focus of the Conference will be on Software Engineering and Software Management. Papers on the following topics are welcomed: 1) Software Engineering and Software Management Practices; 2) Forth Applications; and 3) Forth Technology. Submit a 200-word abstract by March 30 to Lawrence P. Forsely, Laboratory for Laser Energetics, 250 East River Road, Rochester, NY 14623 (716) 235-0168.

DDI

### Reader Ballot

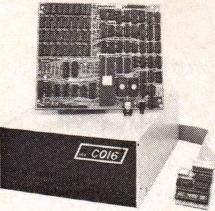
Vote for your favorite feature/article.  
Circle Reader Service No. 198.



**COMPUTEREYES** is a low-cost video acquisition system for the Commodore 64 series. The product includes machine language image capture routines and image save-to-disk capability. **COMPUTEREYES** is the Commodore version of Digital Vision's video acqui-

sition system for the Apple II series. The complete system, including video camera, is priced at \$349.95 from Digital Vision, Inc., 14 Oak Street Suite 2, Needham, MA 02192 (617) 444-9040. **Reader Service No. 113.**

# Z80 POWER<sup>2</sup>



## 16/32 BIT PROCESSING POWER

In 15 minutes you can have a 16 bit O.S. running and still use your CPM80 with the touch of a key.

### WE SUPPORT

**CPM80 RAM DISK**

**MSDOS**

**CPM86**

**CPM68K**

**OS-9\* UNIX Look-A-Like**  
**1.25 Mb RAM\*\***

6 Mhz No Wait States

Real Time Clock

Math Co-Processors

AND MORE

Compatible with any Z80 System Running CPM 2.2 or 3.0.

## HSC's CO-1686 and CO-1668 Attached Resource Processors

Prices Starting  
at \$695.



**Hallock  
Systems  
Company, Inc.**

*Blazing the Trail*

**262 E. Main St.  
Frankfort, NY 13340  
(315) 895-7426**

\* Available First Quarter

\*\* CO-1686 Expandable to 768Kb

Circle no. 26 on reader service card.

# ADVERTISER INDEX

| Reader Service | Page                                 | Reader Service | Page                                   |       |
|----------------|--------------------------------------|----------------|--|-------|
| No.            | Advertiser                           | No.            | Advertiser                             |       |
| 1              | ABCComputing .....                   | 69             | MacTech .....                          | 85    |
| 3              | AFIPS-OAC '85 .....                  | 15             | MicroDynamics Corp. ....               | 69    |
| 5              | Amanuensis, Inc. ....                | 57             | Micro Smith Computer Tech. ....        | 123   |
| 7              | Ampro Computers, Inc. ....           | 87             | Microprocessors Engineering, Ltd. .... | 85    |
| 8              | Apropos Technology .....             | 59             | Microprocessors Unlimited ....         | 117   |
| 10             | Avocet Systems, Inc. ....            | 55             | Mitek .....                            | 77    |
| 11             | B&L Computer Consultants .....       | 75             | Morgan Computing Co. ....              | 117   |
| 13             | B.G. Micro .....                     | 51             | Mullen Computer Products .....         | 11    |
| 12             | BD Software, Inc. ....               | 89             | Nantucket .....                        | 7     |
| 14             | Borland International .....          | C-IV           | Nicolet Paratronics .....              | 65    |
| 19             | cLINE .....                          | 16             | Northwest Computer Algorithms ..       | 69    |
| 15             | C Source .....                       | 65             | OPT-Tech Data Processing .....         | 85    |
| 16             | C Systems .....                      | 103            | Phoenix Computer Products .....        | 9     |
| 17             | C User's Group .....                 | 87             | Poor Persons Software .....            | 83    |
| 18             | C Ware .....                         | 63             | Procode International .....            | 81    |
| 4              | California Digital Engineering ..... | 71             | Programmer's Connection .....          | 113   |
| 20             | Catspaw .....                        | 123            | QCAD Systems Inc. ....                 | 33    |
| 6              | Central Point Software .....         | 73             | Rational Systems, Inc. ....            | 66    |
| 9              | Compucraft .....                     | 123            | Revasco .....                          | 71    |
| 22             | Computer Helper Ind. ....            | 59             | Satellite Software International ..    | C-II  |
| 27             | Creative Solutions .....             | 41             | Selfware, Inc. ....                    | 71    |
| 21             | Cucumber Bookshop .....              | 105            | SemiDisk Systems .....                 | 59    |
| 23             | Cybernetic MicroSystems .....        | 123            | Shaw American Technologies .....       | 61    |
| 28             | D&W Digital .....                    | 28             | Simpliway Products Company .....       | 35    |
| 29             | Data Access Corporation .....        | 3              | Soft Advances .....                    | 65    |
| 30             | Data Base Decisions .....            | 93             | Softaid Inc. ....                      | 123   |
| 31             | Datalight .....                      | 99             | Softfocus .....                        | 123   |
| 32             | Dedicated Microsystems, Inc. ....    | 57             | Software Horizons, Inc. ....           | 81    |
| 33             | Digital Research Computers .....     | 47             | Softway, Inc. ....                     | 81    |
| 34             | Disks Plus .....                     | 123            | Solution Systems .....                 | 111   |
| 24             | Echelon, Inc. ....                   | 30             | Solution Systems .....                 | 111   |
| 35             | Ecosoft, Inc. ....                   | 73             | Solutionware .....                     | 117   |
| *              | Edward Ream .....                    | 35             | Speedware .....                        | 83    |
| 36             | Essential Software .....             | 57             | Spruce Technologies Corp. ....         | 77    |
| 37             | Faircom .....                        | 61             | Starlight Forth Systems .....          | 123   |
| 25             | Flagstaff Engineering .....          | 29             | Stride Micro .....                     | C-III |
| 40             | Fox Software Inc. ....               | 37             | Summit Software .....                  | 109   |
| 42             | GTEK .....                           | 101            | Systems Engineering Tools .....        | 14    |
| 43             | Greenleaf Software Inc. ....         | 96             | The Programmer's Shop .....            | 125   |
| 26             | Hallock Systems Consultants .....    | 128            | Thunder Software .....                 | 128   |
| 44             | Harvard Softworks .....              | 73             | UniPress Software .....                | 13    |
| 46             | IQ Software .....                    | 43             | Vance Info Systems .....               | 119   |
| 38             | ITT Information Systems .....        | 2              | Victory Enterprises Tech. ....         | 119   |
| 48             | Ilyes Systems .....                  | 38             | WL Computer Systems .....              | 121   |
| *              | Integral Quality .....               | 75             | Whitesmith Ltd. ....                   | 1     |
| 50             | Interface Technologies .....         | 24-25          | Wizard Systems Software .....          | 61    |
| 53             | Kimrich Computer Design .....        | 99             | DDJ Subscription .....                 | 83    |
| 54             | Korsmeyer Electronics Design .....   | 75             | DDJ Advertising .....                  | 85    |
| 55             | Laboratory Microsystems Inc. ....    | 121            | DDJ Back Issues .....                  | 97    |
| 57             | Lark Software .....                  | 123            | DDJ Bound Volume .....                 | 114   |
| 58             | Lattice, Inc. ....                   | 77             | DDJ Reprints .....                     | 121   |
| 59             | Leo Electronics, Inc. ....           | 119            |  |       |

## Thunder Software

- **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 operating system. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a 24 page users guide, newsletters. Macro preprocessor, runs on APPLE II+ IIIf //e //c. Source code for libraries is included. Only \$49.95
- **ASSYST: The Assembler System** - A complete 6502 editor/assembler and linker for APPLE DOS 3.3. Menu driven, excellent error trapping, 24 p. users guide, demo programs, source code for all programs! Great for beginners. Only \$23.50
- **THUNDER XREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. Source code and documentation provided. Only \$19.95

Thunder Software POB 31501 Houston Tx 77231 713-728-5501  
Include \$3.00 shipping. COD, VISA and MASTERCARD accepted

Circle no. 102 on reader service card.

# "Despite the recent press notices, multiuser microcomputers aren't anything new!"

**This is the first in a series of discussions with Rod Coleman, President of Stride Micro (formerly Sage Computer) on the 68000 multiuser market and its current environment.**

**Q:** Why do you say that?

**RC:** "The technology to build a high performance multiuser system has been around for five years. And while some of the leaders in this industry have been pretending that micro multiuser didn't exist, we've been shipping complete systems for nearly three years. The benefits of multiuser are undeniable; it is more cost effective, and offers greater flexibility and utility. But until just recently, the marketing pressure to be compatible instead of being better, has blinded the industry."

**Q:** What do you mean?

**RC:** "Well, for example, the Motorola 68000 processor introduced 16/32-bit technology to the personal computer world a long time ago. It was fully capable of

were clearly inferior from a technical point of view. This phenomenon leads me to believe that they will soon rewrite the old proverb: 'Build a better mousetrap and the world will beat a path to your door,' but only if they can find the way through the marketing fog."

**Q:** Are things changing now?

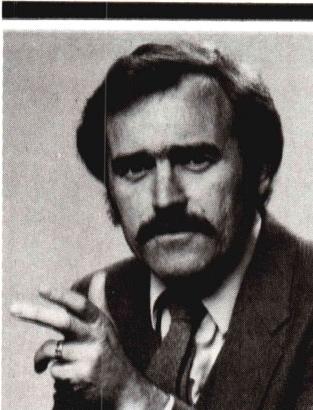
**RC:** "Yes and no. With the business world starting to take more and more interest in microcomputer solutions, the advantages of a solid multiuser system couldn't be kept hidden forever; companies like ours and a few others were beginning to make a dent. Instead of taking a fresh approach, some of the newest multiuser offerings will probably only give the technology an undeserved black eye! Multiuser is far more than the ability to plug in more terminals. It involves things like machine compatibility, fast processors, adequate memory, large storage capacities, backup features, networking, and operating system flexibility."

**Q:** Is this what makes the new Stride 400 Series different?

**RC:** "Exactly. That sounds self-serving, but it's true. Today a number of companies are introducing their first multiuser system. We've been building and shipping multiuser machines for almost three years. We know the pitfalls, we've fallen into some of them. But we have learned from our mistakes."

**Q:** Give me some examples.

**RC:** A hard disk is almost mandatory for any large multiuser installation. Yet, backing up a hard disk can be a nightmare if you only have floppies to work with. That's why we've added a tape backup option to all the larger Stride 400 Series machines. It's irresponsible for a manufacturer to market a multiuser system without such backup. Another good lesson was bus design. We started with one of our own designs, but learned that it's important not only to find a bus that is powerful, but also one that has good support and a strong future to serve tomorrow's needs. We



**"The marketing pressure to be compatible instead of being better, has blinded the industry."**

think the VMEbus is the only design that meets both criteria and thus have made it a standard feature of every Stride 400 Series machine."

**Q:** What are some of the other unique features of the 400 Series?

**RC:** "A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it. Our systems are completely compatible with each other from the 420 model starting at \$2900, through the 440, on to the powerful 460 which tops out near \$60,000. Each system can talk to the others via the standard built-in local area network. Go ahead and compare this with others in the industry. You'll find their little machines don't talk to their big ones, or that the networking and multiuser are incompatible, or that they have different processors or operating systems, and so on."

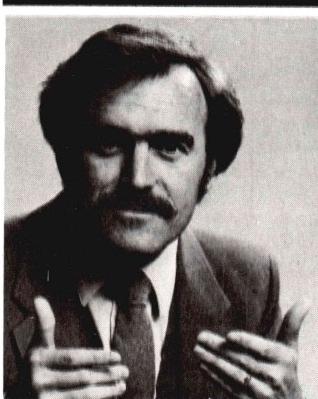
**Q:** When you were still known as Sage Computer, you had a reputation for performance, is that still the case with the new Stride 400 Series?

**RC:** "Certainly, that's our calling card: 'Performance By Design.' Our new systems are actually faster; our standard processor is a 10 MHz 68000 running with no wait

states. That gives us a 25% increase over the Sage models. And, we have a 12 MHz processor as an option. Let me add that speed isn't the only way to judge performance. I think it is also measured in our flexibility. We support a dozen different operating systems, not just one. And our systems service a wide variety of applications from the garage software developer to the corporate consumer running high volume business applications."

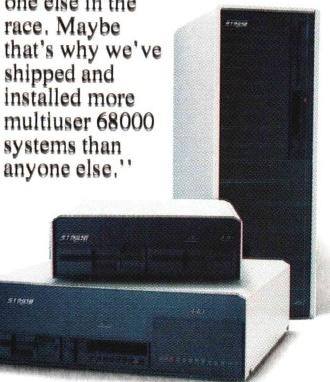
**Q:** Isn't that the same thing all manufacturers say in their ads?

**RC:** "Sure it is. But to use another over used-term, 'shop around'. We like to think of our systems as 'full service 68000 supermicrocomputers.' Take a look at everyone else's literature and then compare. When you examine cost, performance, flexibility, and utility, we don't think there's anyone else in the race. Maybe that's why we've shipped and installed more multiuser 68000 systems than anyone else."



**"A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it."**

meeting high performance and multiuser design requirements in 1980. Instead of this trend taking off, most energy was spent promoting 8088/8086 products that



**STRIDE**  
MICRO

Formerly Sage Computer

For more information on Stride or the location of the nearest Stride Dealer call or write us today. We'll also send you a free copy of our 32 page product catalog.

Corporate Offices:

4905 Energy Way  
Reno, NV 89502  
(702) 322-6868

Regional Offices:

Boston: (617) 229-6868  
Dallas: (214) 392-7070

# NEW from BORLAND!

## TURBO TOOLBOX & TURBO TUTOR

"TURBO is much better than the Pascal IBM sells."  
Jerry Pournelle, Byte, July 1984  
"If you have the slightest interest in Pascal—buy it."  
Bruce Webster, Softalk IBM, March 1984

# BORLAND INTERNATIONAL GIFT PACK

ONLY  
**\$99.95**  
A SAVINGS OF \$30!

What a gift for you and your friends! The extraordinary TURBO PASCAL compiler, together with the exciting new TURBO TOOLBOX and new TURBO TUTOR. All 3 manuals with disks for \$99.95.

**TURBO PASCAL** Version 2.0 (reg. \$49.95). The now classic program development environment still includes the FREE MICROCALC SPREAD SHEET. Commented source code on disk

- Optional 8087 support available for a small additional charge

**NEW! TURBO TOOLBOX** (reg. \$49.95). A set of three fundamental utilities that work in conjunction with TURBO PASCAL. Includes:

- TURBO-ISAM FILES USING B+ TREES. Commented source code on disk
  - QUIK SORT ON DISK. Commented source code on disk
  - GINST (General Installation Program)
- Provides those programs written in TURBO PASCAL with a terminal installation module just like TURBO'S!
- NOW INCLUDES FREE SAMPLE DATABASE

**NEW! TURBO TUTOR** (reg. \$29.95). Teaches step by step how to use the TURBO PASCAL development environment—an ideal introduction for basic programmers. Commented source code for all program examples on disk.

**30 DAY MONEY BACK GUARANTEE**  
These offers good through Feb. 1, 1985

For VISA and MASTERCARD order call toll free:

**1-(800)-255-8008 1-(800)-742-1133**

(Lines open 24 hrs., 7 days a week)

Dealer and Distributor inquiries welcome (408) 438-8400

**CHOOSE ONE** (please add \$5.00 for handling and shipping U.S. orders)

|       |                     |                                 |
|-------|---------------------|---------------------------------|
| _____ | All Three-Gift Pack | \$ 99.95 + 5.00 <b>SPECIAL!</b> |
| _____ | All Three & 8087    | 139.95 + 5.00 <b>SPECIAL!</b>   |
| _____ | Turbo Pascal 2.0    | 49.95 + 5.00                    |
| _____ | Turbo Toolbox       | 49.95 + 5.00                    |
| _____ | Turbo Tutor         | 29.95 + 5.00                    |
| _____ | Turbo 8087          | 89.95 + 5.00                    |

Check \_\_\_\_\_ Money Order \_\_\_\_\_ VISA \_\_\_\_\_ MasterCard \_\_\_\_\_

Card #: \_\_\_\_\_ Exp. date: \_\_\_\_\_ Shipped UPS

My system is: 8 bit \_\_\_\_\_ 16 bit \_\_\_\_\_

Operating System: CP/M 80 \_\_\_\_\_ CP/M 86 \_\_\_\_\_ MS DOS \_\_\_\_\_ PC DOS \_\_\_\_\_

Computer: \_\_\_\_\_ Disk Format: \_\_\_\_\_

Please be sure model number & format are correct.

NAME: \_\_\_\_\_

ADDRESS: \_\_\_\_\_

CITY/STATE/ZIP: \_\_\_\_\_

TELEPHONE: \_\_\_\_\_

California residents add 6% sales tax. Outside U.S.A. add \$15.00 (if outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars). Sorry no C.O.D. or Purchase Orders.

50

**BORLAND**  
INTERNATIONAL

4113 Scotts Valley Drive  
Scotts Valley, CA 95066  
TELEX: 172373

